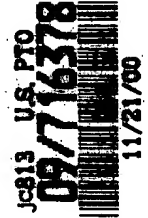


OSP-9705

日 本 国 特 許 庁  
PATENT OFFICE  
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されて  
る事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed  
with this Office.

出 願 年 月 日  
Date of Application:

1999年11月22日

願 番 号  
Application Number:

平成11年特許願第331960号

願 人  
Applicant(s):

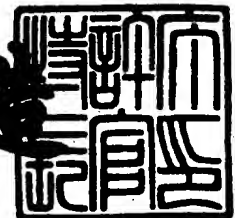
日本電気株式会社

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2000年 7月21日

特許庁長官  
Commissioner,  
Patent Office

及川耕造



出証番号 出証特2000-3058160

【書類名】 特許願

【整理番号】 74310321

【提出日】 平成11年11月22日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 9/00

【発明の名称】 マイクロプロセッサシステム

【請求項の数】 11

【発明者】

【住所又は居所】 東京都港区芝五丁目7番1号 日本電気株式会社内

【氏名】 峰 一雅

【特許出願人】

【識別番号】 000004237

【氏名又は名称】 日本電気株式会社

【代理人】

【識別番号】 100108578

【弁理士】

【氏名又は名称】 高橋 詔男

【代理人】

【識別番号】 100064908

【弁理士】

【氏名又は名称】 志賀 正武

【選任した代理人】

【識別番号】 100101465

【弁理士】

【氏名又は名称】 青山 正和

【選任した代理人】

【識別番号】 100108453

【弁理士】

【氏名又は名称】 村山 靖彦

【手数料の表示】

【予納台帳番号】 008707

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9709418

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 マイクロプロセッサシステム

【特許請求の範囲】

【請求項 1】 プログラムに記述された命令を実行するマイクロプロセッサシステムであって、

第 1 の命令セットをハードウェア上で実行すると共に第 2 の命令セットをソフトウェア上で実行するメインプロセッサと、

前記メインプロセッサの管理下で動作して前記第 2 の命令セットを自発的にフェッチしてハードウェア上で実行するコプロセッサと、

を備えたことを特徴とするマイクロプロセッサシステム。

【請求項 2】 前記コプロセッサは、

前記第 2 の命令セットのうち、前記メインプロセッサの管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサに通知を発行して当該命令の実行を依頼することを特徴とする請求項 1 に記載されたマイクロプロセッサシステム。

【請求項 3】 前記コプロセッサは、

前記第 2 の命令セットのうち、実行頻度の高い所定の複数の命令に対して予め割り付けられた専用割り込みベクタを用いて前記通知を発行することを特徴とする請求項 2 に記載されたマイクロプロセッサシステム。

【請求項 4】 前記コプロセッサは、

前記第 2 の命令セットに属する命令を実行する過程で発生するデータを保持するためのスタックメモリと、該スタックメモリ内の最新データのアドレスを保持するスタックポインタとを有し、前記特定の命令を実行する過程で発生する処理のうち、前記スタックメモリのスタックポインタを更新するための処理を実行するハードウェア資源を備えたことを特徴とする請求項 2 に記載されたマイクロプロセッサシステム。

【請求項 5】 前記コプロセッサは、

現在処理中の前記第 2 の命令セットに属する命令のアドレスを保持するためのプログラムカウンタを有し、前記特定の命令を実行する過程で発生する処理のう

ち、前記プログラムカウンタを更新するための処理を実行するハードウェア資源を備えたことを特徴とする請求項 2 に記載されたマイクロプロセッサシステム。

【請求項 6】 前記コプロセッサは、

前記通知を行う必要が発生したことを表す情報を保持するための状態レジスタを有し、

前記メインプロセッサは、

前記状態レジスタを定期的にアクセスして、該状態レジスタの内容から前記コプロセッサが前記特定の命令に遭遇したことを把握して、前記特定の命令を実行することを特徴とする請求項 2 に記載されたマイクロプロセッサシステム。

【請求項 7】 前記メインプロセッサは、

前記コプロセッサからの前記専用割り込みベクタをエンコードして、処理すべき前記特定の命令に対応する割り込みハンドラを特定する割り込み要求受付回路を備えたことを特徴とする請求項 2 に記載されたマイクロプロセッサシステム。

【請求項 8】 前記メインプロセッサは、

前記コプロセッサの命令キューを参照して、処理すべき前記特定の命令に対応する割り込みハンドラを特定することを特徴とする請求項 2 に記載されたマイクロプロセッサシステム。

【請求項 9】 前記コプロセッサは、

スタックアーキテクチャを有することを特徴とする請求項 1 ないし 8 の何れかに記載されたマイクロプロセッサシステム。

【請求項 10】 前記コプロセッサは、

スタックデータのうち、先頭側の所定数のデータを保持するスタックトップレジスタと、

外部に設けられたスタックメモリと前記スタックトップレジスタとの間に設けられ、前記スタックメモリに保持されたデータの一部をキャッシュするキャッシュメモリと、

を含むことを特徴とする請求項 9 に記載されたマイクロプロセッサシステム。

【請求項 11】 プログラムに記述された複数の処理内容に対応づけて、複数の前記コプロセッサを備えたことを特徴とする請求項 1 ないし 10 の何れかに

記載されたマイクロプロセッサシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明は、プログラムに記述された命令を実行するマイクロプロセッサに関し、特にメインプロセッサとコプロセッサとから構成されるマイクロプロセッサシステムに関する。

【0002】

【従来の技術】

従来、各種の命令セットを実行するための汎用のマイクロプロセッサがある。この汎用のマイクロプロセッサは、ハードウェア上での処理に加えて、ソフトウェア上で適応的に処理する機能を有しているので、J a v a などインタプリタ言語を実行するための命令セットや、他のC P U をエミュレートするための命令セットなどのように、異なる体系の各種の命令セットに柔軟に対応することができる。

【0003】

また、上述のような汎用のマイクロプロセッサをメインプロセッサとし、このメインプロセッサの管理下で動作するマイクロプロセッサをコプロセッサとして備えるマイクロプロセッサシステムがある。この種のマイクロプロセッサシステムでは、コプロセッサは、メインプロセッサの処理の一部を高速化するためのものであるものであって、例えばメインプロセッサ側のソフトウェア上で行われる特定の命令セットを実行するための専用のハードウェアを備えて構成されている。

【0004】

【発明が解決しようとする課題】

ところで、上述の汎用のマイクロプロセッサによれば、柔軟に各種の命令セットを実行することができるものの、例えばJ a v a 言語などの特定のインタプリタ言語の命令を実行する場合、処理に時間を要するという問題がある。つまり、汎用のマイクロプロセッサは、各種の命令セットに柔軟に対応する必要上、ハードウェアのアーキテクチャとして、汎用レジスタアーキテクチャや、アキュムレ

ータアーキテクチャを採用しているのが通例である。このため、ハードウェアとの親和性の観点では、C言語などの高級言語の処理に適している反面、Java言語などのインタプリタ言語についてはソフトウェア上で対処する必要があり、したがって処理に時間を要していた。

## 【0005】

また、メインプロセッサとコプロセッサとから構成された上述のマイクロプロセッサシステムによれば、Java言語などの特定のプログラム言語の命令セットのすべての命令を専用処理するものとしてコプロセッサのハードウェアを構成すれば、この特定の命令セットについては、専用のハードウェア上で実行される。このため、その特定の命令セットを高速に実行することができるものの、コプロセッサの回路規模が大きくなり、したがって全体としてチップ面積が増大するという問題がある。

## 【0006】

この発明は、上記事情に鑑みてなされたもので、回路規模の増大を抑えつつ、特定の命令セットを高速に実行することが可能なマイクロプロセッサシステムを提供することを目的とする。

## 【0007】

## 【課題を解決するための手段】

この発明は、上記課題を解決するため、以下の構成を有する。

すなわち、この発明にかかるマイクロプロセッサシステムは、プログラムに記述された命令を実行するマイクロプロセッサシステムであって、第1の命令セット（例えば後述するプログラムメモリ300に格納されているプログラムの命令）をハードウェア上で実行すると共に第2の命令セット（例えば後述するプログラムメモリ400に格納されているプログラムの命令）をソフトウェア上で実行するメインプロセッサ（例えば後述するメインプロセッサ100に相当する構成要素）と、前記メインプロセッサの管理下で動作して前記第2の命令セットを自発的にフェッチしてハードウェア上で実行するコプロセッサ（例えば後述するコプロセッサ200に相当する構成要素）と、を備えたことを特徴とする。

## 【0008】

この構成によれば、第2の命令セットについては、メインプロセッサまたはコプロセッサにより適応的に処理することができる。したがって、例えば、原則として第2の命令セットの処理をコプロセッサのハードウェア上で処理することとし、例外として第2の命令セットの内のソフトウェアによる処理に適した命令のみをメインプロセッサで実行するようにすれば、マイクロプロセッサシステムに要求される処理能力に対して、全体としてハードウェアの規模を縮小される。したがって、回路規模の増大を抑えつつ、特定の命令セット（第2の命令セット）を高速に実行することが可能となる。

## 【0009】

また、前記コプロセッサは、命令フェッチ回路とプログラムカウンタと命令キューとを有し、命令フェッチ回路が、プログラムカウンタの値を受け取り、この値をアドレスとして使用してプログラムメモリにアクセスし、得られたデータを命令キューにセットすることを特徴とする。

## 【0010】

この構成によれば、コプロセッサが自発的に命令をフェッチして実行することができ、コプロセッサが、第2の命令セットをハードウェアで実行することができる。こうすることで、メインプロセッサがプログラムメモリから第2の命令セットをデータとしてフェッチし、コプロセッサの命令キューにセットする、という動作を実行せずに済み、メインプロセッサの負担を軽減できるとともに、フェッチの動作に要する時間が短くなり、第2の命令の実行速度の向上が実現でき、システム全体の処理性能の向上が実現可能となる。

## 【0011】

また、前記コプロセッサは、例えば、前記第2の命令セットのうち、前記メインプロセッサの管理下にあるデータ（例えば後述するデータメモリ500に格納されているデータ）を操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサに通知を発行して当該命令の実行を依頼することを特徴とする。

## 【0012】

この構成によれば、メインプロセッサの管理下にあるデータを操作する必要を生ずる特定の命令については、メインプロセッサのソフトウェア上で、このメイ



ンプロセッサの管理下にあるデータを操作して処理される。換言すれば、ソフトウェアによる処理に適した命令や、コプロセッサでは処理不能な命令については、メインプロセッサのソフトウェア上で処理される。これにより、メインプロセッサのソフトウェアが有効に活用されると共にコプロセッサのハードウェアの規模を必要最小限に抑えることが可能となり、したがって全体としてハードウェアの規模を有効に抑えることが可能となる。

## 【0013】

さらに、前記コプロセッサは、例えば、前記第2の命令セットのうち、実行頻度の高い所定の複数の命令に対して予め割り付けられた専用割り込みベクタ（例えば後述する専用割り込み要求信号S290A～S290Fにより指定される専用割り込みベクタに相当する要素）を用いて前記通知を発行することを特徴とする。

## 【0014】

この構成によれば、第2の命令セットのうち、実行頻度の高い命令に遭遇した場合、この命令に割り付けられた専用割り込み要求信号を用いてメインプロセッサへの通知が行われる。メインプロセッサは、この専用割り込み要求信号により、コプロセッサからの通知に加えて、コプロセッサが遭遇した命令が把握される。したがって、メインプロセッサ側において、コプロセッサからの依頼により処理すべき命令を識別するための処理を省くことが可能となる。

## 【0015】

さらにまた、前記コプロセッサは、例えば、前記第2の命令セットに属する命令を実行する過程で発生するデータを保持するためのスタックメモリ（例えば後述するスタックメモリ270に相当する構成要素）と、該スタックメモリ内の最新データのアドレスを保持するスタックポインタ（例えば後述するスタックポインタ260Aに相当する構成要素）とを有し、前記特定の命令を実行する過程で発生する処理のうち、前記スタックメモリのスタックポインタを更新するための処理を実行するハードウェア資源（例えば後述するデコーダ240A、処理手順制御信号発生回路240B、SP用増減値発生回路240C、加算器250Aに相当する構成要素）を備えたことを特徴とする。

## 【0016】

この構成によれば、メインプロセッサに処理を依頼する命令であっても、この命令を実行する過程で発生する処理のうち、スタックポインタを更新するための処理については、コプロセッサのハードウェア上で行われる。したがって、メインプロセッサが依頼を受けてこの命令を処理する過程において、スタックポインタを更新するための処理を省くことができる。このため、メインプロセッサのソフトウェア上での処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を短縮することができる。

## 【0017】

さらにまた、前記コプロセッサは、例えば、現在処理中の前記第2の命令セットに属する命令のアドレスを保持するためのプログラムカウンタ（例えば後述するプログラムカウンタ260Bに相当する構成要素）を有し、前記特定の命令を実行する過程で発生する処理のうち、前記プログラムカウンタを更新するための処理を実行するハードウェア資源（例えば後述するデコーダ240A、処理手順制御信号発生回路240B、PC用増減値発生回路240D、加算器250Bに相当する構成要素）を備えたことを特徴とする。

## 【0018】

この構成によれば、メインプロセッサに処理を依頼する命令であっても、この命令を実行する過程で発生する処理のうち、プログラムカウンタを更新するための処理については、コプロセッサのハードウェア上で行われる。したがって、メインプロセッサが依頼を受けてこの命令を処理する過程において、プログラムカウンタを更新するための処理を省くことができる。このため、メインプロセッサのソフトウェア上での処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を短縮することができる。

## 【0019】

さらにまた、前記コプロセッサは、例えば、前記通知を行う必要が発生したことを表す情報を保持するための状態レジスタを有し、前記メインプロセッサは、

前記状態レジスタを定期的にアクセスして、該状態レジスタの内容から前記コプロセッサが前記特定の命令に遭遇したことを把握して、前記特定の命令を実行することを特徴とする。

この構成によれば、メインプロセッサが定期的に状態レジスタをアクセスして、コプロセッサが特定の命令に遭遇したことを把握するので、メインプロセッサに対し、割り込み要求信号を用いて割り込み処理を行う必要がなくなる。したがって、コプロセッサからメインプロセッサへの通知を簡略化することが可能となる。

#### 【0020】

さらにまた、前記メインプロセッサは、例えば、前記コプロセッサからの前記専用割り込みベクタをエンコードして、処理すべき前記特定の命令に対応する割り込みハンドラを特定する割り込み要求受付回路（例えば後述する割り込み要求受付回路190に相当する構成要素）を備えたことを特徴とする。

この構成によれば、メインプロセッサにおいて、コプロセッサから入力する専用割り込みベクタに基づいて、直接的に割り込みハンドラが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウェア上の処理が不要となり、メインプロセッサでの処理時間を短縮することが可能となる。

#### 【0021】

さらにまた、前記メインプロセッサは、例えば、前記コプロセッサの命令キューを参照して、処理すべき前記特定の命令に対応する割り込みハンドラを特定することを特徴とする。

この構成によれば、メインプロセッサにおいて、コプロセッサの命令キューを参照してコプロセッサが遭遇した命令が把握され、直接的に割り込みハンドラが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウェア上の処理が簡略化され、メインプロセッサでの処理時間を短縮することが可能となる。

#### 【0022】

さらにまた、前記コプロセッサは、スタックアーキテクチャ（例えば後述する

コプロセッサ 200, 200A, 200B が採用するアーキテクチャに相当する要素) を有することを特徴とする。

この構成によれば、コプロセッサにおいて、サブルーチンの頻繁な使用を伴う命令や、ゼロオペランドのアドレッシングを伴う命令の処理が効率的に行われる。したがって Java などのスタックアーキテクチャのインタプリタ言語の命令を効率的に実行することが可能となる。

#### 【0023】

さらにまた、スタックアーキテクチャを採用する前記コプロセッサは、例えば、スタックデータのうち、先頭側の所定数のデータを保持するスタックトップレジスタ (例えば後述するスタックトップレジスタ 270A に相当する構成要素) と、外部に設けられたスタックメモリと前記スタックトップレジスタとの間に設けられ、前記スタックメモリに保持されたデータの一部をキャッシュするキャッシュメモリ (例えば後述すキャッシュメモリ 270C に相当する構成要素) と、を含むことを特徴とする。

この構成によれば、スタックデータのうち、先頭側の所定数のデータをアクセスする頻度の高いプログラム言語の命令を効率的に実行することができると共に、スタックメモリの容量を適応的に拡大することが可能となる。

#### 【0024】

さらにまた、プログラムに記述された複数の処理内容に対応づけて、複数の前記コプロセッサ (例えば後述するコプロセッサ 200X ~ 200Z に相当する構成要素) を備えたことを特徴とする。

この構成によれば、プログラム中の複数の処理を並列処理することが可能となり、処理を高速化することができる。

#### 【0025】

##### 【発明の実施の形態】

次に、この発明の実施の形態について、図面を参照して詳細に説明する。

##### <実施の形態 1>

図 1 に、この発明の実施の形態 1 にかかるマイクロプロセッサシステムの構成を示す。このマイクロプロセッサシステムは、異なるプログラム言語で記述され

た複数種類の命令セットを実行可能なように構成されたものであって、いわゆる汎用レジスタ・アーキテクチャを採用する汎用のマイクロプロセッサ 1 0 0 と、スタック・アーキテクチャを採用するコプロセッサ 2 0 0 とを主要な構成要素として備える。

## 【 0 0 2 6 】

メインプロセッサ 1 0 0 は、C 言語やアセンブラなどのプログラム言語で記述された第 1 の命令セットをハードウェア上で実行し、J a v a 言語などの特定のプログラム言語で記述された第 2 の命令セットをソフトウェア上で実行するものである。また、コプロセッサ 2 0 0 は、メインプロセッサ 1 0 0 の管理下で協調動作するものであって、上述の第 1 の命令セットとは異なる J a v a などの第 2 の命令セットをハードウェア上で実行するものである。

## 【 0 0 2 7 】

プログラムメモリ 3 0 0 は、メインプロセッサ 1 0 0 の一連の動作を制御するためのプログラムであって第 1 の命令セットの機械語を格納する。プログラムメモリ 4 0 0 は、J a v a などの第 2 の命令セットの中間コードを格納する。データメモリ 5 0 0 は、メインプロセッサ 1 0 0 に入力される演算データやその演算結果など、メインプロセッサ 1 0 0 で操作されるデータを格納する。

この実施の形態 1 では、プログラムメモリ 3 0 0 には、C 言語で記述されたプログラムの機械語であってメインプロセッサの動作手順が記述された機械語が展開されているものとし、プログラムメモリ 4 0 0 には、J a v a 言語などのインタプリタ言語の中間コードが展開されているものとする。

## 【 0 0 2 8 】

これらプログラムメモリ 3 0 0、4 0 0、およびデータメモリ 5 0 0 は、外部バス 6 0 0 を介して互いに接続されると共にメインプロセッサ 1 0 0 に接続される。上述のメインプロセッサ 1 0 0、コプロセッサ 2 0 0、プログラムメモリ 3 0 0、4 0 0、データメモリ 5 0 0、および外部バス 6 0 0 は、同一プリント基板上に実装されてマイクロプロセッサシステムを構成する。もしくは、これらの一部あるいは全部を 1 つのシステム L S I 上に集積してもよい。

## 【 0 0 2 9 】

ここで、メインプロセッサ 100 は、プログラム中の第 2 の命令セットを解読してプログラムの進行を制御するための回路系として、プログラムメモリ 300 をアクセスして第 1 の命令セットに属する命令（以下、単に第 1 の命令と称す）をフェッチする命令フェッチ回路 110、フェッチされた第 1 の命令を一時的に保持する命令キュー 120、第 1 の命令を解読する命令デコーダ 130、メインプロセッサ側が現在処理中の命令のアドレス（即ちプログラムカウンタの値、以下 PC 値と称す）を更新する加算器 140、PC 値を保持するプログラムカウンタ 150 を有する。

#### 【0030】

また、このメインプロセッサ 100 は、命令デコーダ 130 の解読結果に応じた演算を実行するための回路系として、この演算の入力データや演算結果（出力データ）を一時的に保持する汎用レジスタ 160、この入力データに対して演算処理を施す算術論理ユニット（ALU）170、この算術論理ユニット 170 により生成されたアドレスを使用して汎用レジスタ 160 と外部のデータバス 600 との間でデータをロードまたはストアするデータアクセス回路 180、および後述のコプロセッサ 200 からの割り込み要求を受け付けるための割り込み要求受付回路 190 を有する。

さらに、この汎用のメインプロセッサ 100 は、第 1 の命令セットをハードウェア上で実行することにより、第 2 の命令セットをソフトウェア上で間接的に実行可能なように構成されている。

#### 【0031】

一方、コプロセッサ 200 は、第 2 の命令セットのうち、メインプロセッサ 100 の管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、メインプロセッサ 100 に通知して、この命令の実行を依頼するものである。

具体的には、コプロセッサ 200 は、プログラム中の第 2 の命令セットを解読してプログラムの進行を制御するための回路系として、このコプロセッサ自体の特定の基本動作を制御するための制御データ（例えば動作開始や動作停止を指示するデータ）を一時的に保持するの制御レジスタ 210、制御レジスタ 210 に保持された制御データに応じてプログラムメモリ 400 から第 2 の命令セットに

属する命令（以下、単に第2の命令と称す）をフェッチするための命令フェッチ回路220、フェッチされた第2の命令を一時的に保持する命令キュー230、フェッチされた第2の命令を解読する命令デコーダ240、スタックされたデータの先頭アドレス（即ちスタックポインタの値、以下、SP値と称す）を命令デコーダの指示により更新するための加減算器250A、SP値を保持するスタックポインタ260A、現在処理中の命令のアドレス（即ちプログラムカウンタ値、以下PC値と称す）を命令デコーダの指示により更新するための加算器250B、PC値を保持するプログラムカウンタ260Bを有する。

なお、この発明において、「スタックされたデータの先頭アドレス」、すなわち「SP値」は、スタックされた最新データのアドレスを意味するものとする。

#### 【0032】

また、このコプロセッサ200は、命令デコーダ240の解読結果に応じた演算を実行するための回路系として、第2の命令セットに属する命令を実行する過程で発生する各種のデータ（第2の命令の演算オペランド、ローカル変数、手続オペランド等）を保持するスタックメモリ270、このスタックメモリ上のオペランドに演算処理を施す算術論理ユニット（ALU）280、および命令デコーダ240の解読結果に応じてメインプロセッサ100に対し割り込み要求を発生する割り込み要求発生回路290を有する。なお、スタックメモリ270でのスタックは、アドレスの小さい方から大きい方に成長するものとし、スタックのデータ幅は例えば32ビット（4バイト）である。

#### 【0033】

これら、制御レジスタ210、スタックポインタ260A、プログラムカウンタ260B、およびスタックメモリ270は、コプロセッサ200の内部バス201を介して外部のデータバス600に接続されており、メインプロセッサ100によりコプロセッサ側の制御レジスタ210やスタックポインタ260Aの内容を直接的に設定可能なように構成されている。

#### 【0034】

ここで、図2を参照して、スタックメモリ270の動作概念を補足説明しておく。

このスタックメモリ270は、LIFO（後入れ先出し）方式のメモリであって、プログラムの実行中にデータを順々に積み重ねるようにして一時的に保持するものである。図2に示す例は、スタックポインタが示す先頭アドレスおよびその前のアドレスにそれぞれ収納された二つのデータに対して二項演算を施す場合におけるスタックメモリの内容の変化の様子を示す。

## 【0035】

演算に先だって、同図（a）に示すように、アドレス「0x1004」～「0x1014」には、データD1～D5が順次積み重ねられるように収納される。この場合、最後に収納されたデータD5のアドレス「0x1014」が先頭アドレスとなり、スタックポインタの値とされる。この状態から、二項演算（例えば二項の加算演算）が実行されると、スタックポインタで示されるアドレス「0x1014」のデータD5とアドレス「0x1010」のデータD4とに対して演算が行われる。

## 【0036】

具体的には、この演算の内容が加算演算であれば、データD4とデータD5とを加算して、その演算結果が算出される。この時点で、データD4、D5は不要となる。そこで、図2（b）に示すように、演算結果として算出されたデータD6は、後続のデータD3の次のアドレス「0x1010」に保存され、スタックポインタの値がアドレス「0x1010」に更新される。

このように、スタックメモリには、演算を処理する過程で一時的にデータが保持され、スタックポインタによりその時々で有効なデータの先頭アドレスが指定されるようになっている。

## 【0037】

図3に、命令デコーダ240の詳細な構成とその周辺を示す。

整列回路221は、命令フェッチ回路220の一部機能として構成されたもので、フェッチされた第2の命令から、オペレーションコードとオペランドコードとを抽出し、これらを整列して命令キュー230に出力する。

命令デコーダ240は、命令キュー230に保持されたオペレーションコードを解読するためのデコーダ240A、オペレーションコードの解読結果に応じた



処理手順制御信号 S 2 4 0 B を発生する処理手順制御信号発生回路 2 4 0 B、S P 値の増減値を発生する S P 用増減値発生回路 2 4 0 C、および P C 値の増減値を発生する P C 用増減値発生回路 2 4 0 D から構成される。

## 【 0 0 3 8 】

図 4 に、メインプロセッサ 1 0 0 内の割り込み要求受付回路 1 9 0 の詳細と、コプロセッサ 2 0 0 内の割り込み要求発生回路 2 9 0 の詳細を示す。

割り込み要求発生回路 2 9 0 は、割り込み処理用デコーダ 2 9 1 と、複数のフリップフロップ 2 9 2 とから構成される。割り込み処理用デコーダ 2 9 1 は、命令デコーダ 2 4 0 からの割り込み要求信号 S 2 4 1 A に基づき、複数の専用割り込み要求信号 S 2 9 0 A ～ S 2 9 0 F と、共用の割り込み要求信号 S 2 9 0 G とから構成される割り込みベクタ S 2 9 0 を出力する。以下の説明において、適宜、複数の専用割り込み要求信号 S 2 9 0 A ～ S 2 9 0 F を「専用割り込みベクタ」と称す。

## 【 0 0 3 9 】

ここで、専用割り込み要求信号 S 2 9 0 A ～ S 2 9 0 F は、第 2 の命令セットのうち、コプロセッサが単独で処理不能な命令であって、比較的実行頻度の高い特定の 1 つ又は複数の命令に対して予め割り付けられている。つまり、コプロセッサがこの特定の命令に遭遇した場合、この遭遇した命令に割り付けられた専用割り込み要求信号が活性化されることにより、メインプロセッサは、この活性化された専用割り込み要求信号から、遭遇した命令を直接的に把握可能なようになっている。また、共用の割り込み要求信号 S 2 9 0 G は、コプロセッサが単独で処理不能な命令であって上述の特定の命令以外の命令に共通に割り付けられている。

このように、コプロセッサ 2 0 0 は、第 2 の命令セットのうち、実行頻度の高い 1 つ又は複数の命令に対して予め割り付けられた 1 つ又は複数の専用割り込み要求信号を用いて「通知」を発行するように構成されている。

## 【 0 0 4 0 】

この実施の形態 1 では、コプロセッサ側で単独処理可能か否かの判断は、命令を処理（実行）する過程で、メインプロセッサ 1 0 0 の管理下にあるデータ（例

例えばデータメモリ 500 に格納されたデータ) を操作する必要を生ずるか否かを基準として行われる。つまり、コプロセッサ 200 は、メインプロセッサ側の管理下にあるデータを操作する必要を生じる命令については、単独処理不能と判断し、これ以外の命令については単独処理可能と判断する。コプロセッサ 200 側で単独処理不能な特定の命令については、予め命令デコーダ 240 の内部に定義されている。

#### 【0041】

これら専用割り込み要求信号 S290A~S290F および共用の割り込み要求信号 S290G は、複数のフリップフロップ (F/F) 292 に保持された後、割り込みベクタ S290 としてメインプロセッサ 100 側に出力される。この図 4 に示す例では、命令デコーダ 240 からの 3 種類の割り込み要求信号 S241A の論理値の組み合わせにより第 2 の命令が特定され、この命令の内容に応じて専用割り込み要求信号 S290A~S290G の何れかが選択的に活性化されて出力される。

#### 【0042】

なお、専用割り込み要求信号 S290A~S290F にプライオリティを設定してもよい。この場合、例えば、プライオリティの高い割り込み要求信号には、一つの命令が割り付けられ、プライオリティの低い割り込み要求信号には、複数の命令が割り付けられる。これにより、高いプライオリティを有する割り込み要求信号に割り付けられた命令については、即座に専用のハンドラが特定される。また、プライオリティの低い割り込み要求信号に割り付けられた命令については、後述するように、この割り込み要求信号に割り付けられた複数の命令で共有されるハンドラによって、実際にフェッチされた命令を特定するための処理が行われた後、この命令の割り込みを処理するためのルーチンが特定される。

#### 【0043】

一方、メインプロセッサ側の割り込み要求受付回路 190 は、複数のフリップフロップ 191 と、割り込みハンドラアドレス発生回路 192 と、割り込みアドレスレジスタ 193 とから構成される。コプロセッサ側からの専用割り込み要求信号 S290A~S290F および共用の割り込み要求信号 S290G は、複

数のフリップフロップ 1 9 1 にそれぞれ保持される。割り込みハンドラアドレス発生回路 1 9 2 は、コプロセッサ側からの専用割り込み要求信号 S 2 9 0 A ~ S 2 9 0 G をエンコードして割り込みハンドラのアドレスを発生する。この割り込みハンドラのアドレスは、割り込みアドレスレジスタ 1 9 3 に保持された後に、上述の命令フェッチ回路 1 1 0 に出力される。

#### 【 0 0 4 4 】

このように、割り込み要求受付回路 1 9 0 は、コプロセッサ 2 0 0 からの専用割り込み要求信号 S 2 9 0 A ~ S 2 9 0 F (専用割り込みベクタ) をエンコードして、処理すべき特定の命令に対応する割り込みハンドラを特定するように構成される。

なお、メインプロセッサ 1 0 0 は、コプロセッサ 2 0 0 に対し、その動作を管理するための各種の操作を行う。この操作は、プログラムメモリ 3 0 0 に格納されたプログラムをメインプロセッサ 1 0 0 で実行することにより、ソフトウェア上で行われる。

#### 【 0 0 4 5 】

以下、図 5 および図 6 に示すフローチャートに沿って、J a v a 言語などのスタック・アーキテクチャ命令で記述されたプログラムを実行する場合を例とし、この実施の形態 1 にかかるマイクロプロセッサシステムの動作を説明する。

ステップ S T 1 0 : 最初に、このマイクロプロセッサシステムのリセットが行われる。これにより、各部の動作が初期化され、このリセット後にメインプロセッサ 1 0 0 が動作を開始し、コプロセッサ 2 0 0 が停止状態に置かれる。

#### 【 0 0 4 6 】

ステップ S T 1 1 : 続いて、メインプロセッサ 1 0 0 は、ソフトウェア上での操作により、スタックの先頭アドレスをコプロセッサ 2 0 0 内のスタックポインタ 2 6 0 A に設定し、コプロセッサ 2 0 0 が実行すべきプログラムの先頭アドレスをプログラムカウンタ 2 6 0 B に設定する。これにより、コプロセッサ 2 0 0 のスタックポインタ 2 6 0 A およびプログラムカウンタ 2 6 0 B に初期値がそれぞれ設定される。

#### 【 0 0 4 7 】

ステップST12：続いて、メインプロセッサ100は、ソフトウェア上での操作により、コプロセッサ200内の制御レジスタ210に、「動作開始」を指示するための制御データ（例えばデータ「0」）を設定し、このコプロセッサ200の動作を開始させる。

【0048】

ステップST13：続いて、動作を開始したコプロセッサ200は、ハードウェア上での操作により、プログラムカウンタ260Bに設定されたアドレス（PC値）を参照して、第2の命令セットの中間コードが展開されたプログラムメモリ400をアクセスし、このアドレスから命令フェッチ回路220により自発的に第2の命令をフェッチする。フェッチされた命令は命令キュー230に一旦保持される。

【0049】

ステップST14：続いて、コプロセッサ200は、フェッチされた第2の命令を命令デコーダ240にて解釈（デコード）する。

ステップST15：続いて、コプロセッサ200は、フェッチされた第2の命令がコプロセッサ側で単独処理可能か否かを判断する。

【0050】

ここで、コプロセッサ200が遭遇した第2の命令が、例えばJava仮想マシンの「iadd」（整数のデータ同士を加算する命令）のように、メインプロセッサの管理下に有るデータを操作する必要のない命令、すなわちコプロセッサ側で処理可能な命令である場合（ステップST15：YES）、コプロセッサ200は、この命令を処理するための以下のステップST16～ST19を実行する。

【0051】

ステップST16：コプロセッサ200は、ハードウェア上での操作により、スタックメモリ270から、演算処理の対象とされるデータを読み出す。上述の「iadd」命令の場合、SP値が表すアドレスと、このSP値が表すアドレスから「4」を減算したアドレス（すなわち「SP値-4」）から、演算処理の対象とされるデータをそれぞれ読み出す。

## 【0052】

ステップST17：続いて、コプロセッサ200は、ハードウェア上での操作により、スタックメモリ270から読み出したデータの演算を実行する。具体的には、上述の「i a d d」命令の場合、SP値が表すアドレスから読み出したデータと、「SP値-4」が表すアドレスから読み出したデータとを加算する。

## 【0053】

ステップST18：続いて、コプロセッサ200は、ハードウェア上での操作により、スタックメモリ270に演算結果を書き戻す。上述の「i a d d」命令の場合、演算結果は、SP値が表すアドレスから「4」を減算したアドレス（すなわち「SP値-4」）に書き戻される。

## 【0054】

ステップST19：続いて、コプロセッサ200は、ハードウェア上での操作により、現在処理の対象とされている命令の種類に応じて、スタックポインタ260A内のSP値と、プログラムカウンタ260B内のPC値とを更新する。

## 【0055】

具体的には、命令デコーダ240内のデコーダ240Aの解読結果に応じて、SP用増減値発生回路240Cは、現在のSP値に対する増減値を算出する。加減算器250Aは、算出された増減値を現在のSP値に加算してSP値を更新し、これをスタックポインタ260Aに設定する。また、命令デコーダ240内のデコーダ240Aの解読結果に応じて、PC用増減値発生回路240Dは、現在のPC値に対する増減値を算出する。加算器250Bは、算出された増減値を現在のPC値に加算してPC値を更新し、これをプログラムカウンタ260Bに設定する。

## 【0056】

上述の「i a d d」命令の場合、プログラムカウンタ260Bの現在のPC値に「1」を加算し、スタックポインタ260Aの現在のSP値から「4」を減算して、PC値とSP値を更新する。

この後、上述のステップST13に戻り、次の第2の命令をフェッチして、この命令の内容に応じた処理が同様に行われる。

## 【0057】

次に、フェッチされた第2の命令が、コプロセッサ200側で処理不能な命令である場合の処理を説明する。

コプロセッサ200は、遭遇した第2の命令が、例えばJava仮想マシンの「i a l o a d」命令（整数の配列からデータをロードする命令）のように、メインプロセッサの管理下に有るデータを操作する必要のある命令、即ちコプロセッサが単独では処理不能な命令である場合、上述のステップST15において否定的な判断を下す（ステップST15：NO）。この場合、メインプロセッサ100またはコプロセッサ200において、この命令を処理するための以下のステップST20～ST26が実行される。

## 【0058】

ステップST20：コプロセッサ200は、当該命令は専用のハンドラを持つか否か、すなわち当該命令に対して割り込み処理用のハンドラ（割り込みハンドラ）を特定するための専用割り込み要求信号が割り付けられているか否かを判断する。この実施の形態では、「i a l o a d」命令に対して、専用割り込み要求信号S290Aが割り付けられており、「i a l o a d」命令は専用のハンドラを持つものとする。

## 【0059】

ステップST21：この場合（ステップST20：YES）、コプロセッサ200は、当該命令に割り付けられた専用割り込み要求信号S290Aにより割り込みを発生させ、単独では処理不能な「i a l o a d」命令に遭遇したことをメインプロセッサ100側に通知する。

ステップST22：続いて、コプロセッサ200は、ハードウェア上での操作により、制御レジスタ210に「動作停止」を指示するための制御データ（例えばデータ「1」）を設定し、動作を一旦停止する。

## 【0060】

ステップST23：続いて、専用割り込み要求信号S290Aによりコプロセッサ200から通知を受けたメインプロセッサ100は、処理すべき命令に対応する割り込みハンドラの特定を行う。具体的には、割り込みハンドラアドレス発

生回路 192 によりコプロセッサ 200 からの割り込み要求信号 S290 (S290A~S290G) をエンコードして割り込みハンドラのアドレス (割り込みアドレス) を発生する。

【0061】

ステップ ST24 : 命令が特定されると、続いて、メインプロセッサ 100 は、ソフトウェア上での操作により、スタックメモリ 270 またはデータメモリ 500 から命令の実行 (演算) に必要なデータを読み出して、命令の内容に応じた演算処理を実行する。例えば上述の「i a l o a d」命令の場合、SP 値が表すアドレスで指定されるデータ (インデックス) と、この SP 値が表すアドレスから「4」を減算したアドレス (すなわち「SP 値-4」) で指定されるデータ (配列リファレンス) を、スタックメモリ 270 からそれぞれ読み出す。そして、読み出された配列リファレンスで示される配列中のデータであって、インデックスに該当するデータをデータメモリ 500 から読み出す。

【0062】

ステップ ST25 : 続いて、メインプロセッサ 100 は、ソフトウェア上での操作により、コプロセッサ 200 内のスタックメモリ 270 またはデータメモリ 500 に演算結果を書き戻す。上述の「i a l o a d」命令の場合、メインプロセッサは、SP 値から「4」を減算したアドレス (すなわち「SP 値-4」) を指定して、スタックメモリ 270 に、データメモリ 500 から読み出した上述のデータを書き込む。

【0063】

ステップ ST26 : 続いて、メインプロセッサ 100 は、ソフトウェア上での操作により、コプロセッサ 200 内の制御レジスタ 210 に、「動作開始」を指示するための制御データ (例えばデータ「0」) を設定し、コプロセッサ 200 の動作を再開させる。

この後、上述のステップ ST19 に移行して PC 値と SP 値とを更新した後、ステップ ST13 に戻り、次の命令をフェッチして同様の処理が行われる。

【0064】

次に、フェッチされた命令が、専用のハンドラを持たない場合の処理を説明す

る。

この場合、コプロセッサ200は、上述のステップST20において否定的判断を下す（ステップST20:NO）。この場合、共用の割り込み要求信号S290Gにより通知が行われるが、複数の命令に共有される割り込み要求信号S290Gから、命令の種類を一義的に特定することはできない。そこでこの場合、メインプロセッサ100またはコプロセッサ200において、この命令を処理するための以下のステップST30～ST34、ST24～ST26が実行され、命令を特定するための処理が行われる。

【0065】

ステップST30：この場合、コプロセッサ200は、ハードウェア上での操作により、共用の割り込み要求信号S290Gを活性化して、これをメインプロセッサ100に出力する。

ステップSY31：続いて、コプロセッサ200は、ハードウェア上での操作により、制御レジスタ210に、「動作停止」を指示するための制御データ（例えばデータ「1」）を設定し、その動作を停止する。

【0066】

ステップST32：続いて、共用割り込み要求信号S290Gによりコプロセッサ200から通知を受けたメインプロセッサ100は、共用の割り込みハンドラを特定し、割り込み処理を行う。すなわち、メインプロセッサ100は、ソフトウェア上での操作により、プログラムカウンタ260BのPC値を読み出し、現在処理中の命令のアドレスを把握する。

ステップST33：続いて、メインプロセッサ100は、PC値から把握されたアドレスを指定して、プログラムメモリ400から命令のコードを読み出し、このコードから現在処理中の命令の種類を特定する。

【0067】

ステップST34：続いて、メインプロセッサ100は、特定された命令の種類から、この命令に対応する共用の割り込みハンドラ中の処理ルーチンを特定する。

ステップST24～ST26：この後、メインプロセッサ100は、上述と同



様に、特定された割り込みハンドラにより割り込み処理を行い、特定された命令に応じた演算を実行して結果を必要に応じてデータメモリ 500 等へ書き戻し、コプロセッサ 200 の動作を再開させる。そして、上述のステップ ST 19 に処理を移行させて PC 値および SP 値を更新した後、処理を上述のステップ ST 13 に戻し、次の命令に対する処理を同様に実行する。

#### 【0068】

以上のように、この実施の形態 1 では、コプロセッサ 200 が、フェッチされた命令が単独で処理可能か否かを自ら判断する。そして、コプロセッサが単独では処理不能な命令に遭遇した場合、この命令の演算がメインプロセッサ 100 のソフトウェア上で処理され、この命令の実行に伴って発生する PC 値および SP 値の更新処理がコプロセッサ 200 のハードウェア上で処理される。これにより、メインプロセッサ 100 側では、メインプロセッサがコプロセッサから依頼された命令をソフトウェア上で処理する際、PC 値や SP 値の更新処理を行う必要がなくなるので、インプロセッサ側のソフトウェアによる処理の負荷を軽減することができる。

#### 【0069】

また、この実施の形態では、コプロセッサ 200 が、単独では処理不能な命令に遭遇した場合、この命令に専用に割り付けられた割り込みベクタを用いてメインプロセッサ 100 に通知する。これにより、メインプロセッサ 100 において、割り込みベクタから直接的に命令の種類を把握することができる。したがって、割り込みハンドラを特定する際、ソフトウェア上の処理の負荷を軽減することができる。

#### 【0070】

ここで、1 命令あたりの動作クロック数が 40 クロックと仮定すると、その内の 10 数クロックが、命令の種類を特定するための処理に費やされ、命令を特定するための処理が、大きな負荷となる。したがって、上述のように、専用の割り込みベクタを設けることにより、割り込み処理の際に行われるインタプリタのデコードやプログラムカウンタの読み出しなど、命令の種類を特定するための処理が不要となり、メインプロセッサの処理上の負荷が大幅に軽減される。

## 【0071】

上述の実施の形態1では、専用割り込み要求信号S290A～S290Fが、コプロセッサ側で単独処理不能な命令であって比較的実行頻度の高い特定の複数の命令に対して予め割り付けられるものとしたが、これに限定されることなく、ひとつの専用割り込み要求信号を2以上の命令に割り付けるようにしてもよい。

## 【0072】

ここで、例えば専用割り込み要求信号S290Aを「iadd」命令および「iaload」命令に割り付けた場合、この専用割り込み要求信号S290Aから、コプロセッサが遭遇した命令を一義的に特定することはできない。そこで、この場合、共用の割り込み要求信号S290Gにより割り込み要求がなされた場合と同様に、コプロセッサ200からプログラムカウンタのPC値を読み取って命令の種類を特定する処理（ステップST31～ST34）を行えばよい。このようにひとつの専用割り込み要求信号を2以上の命令に割り付けると、PC値から命令を特定するための処理が必要となるが、より多くの種類の命令に対して専用割り込み要求信号を割り付けることができる。

## 【0073】

また、この実施の形態1では、コプロセッサ200が単独で処理できない命令として、メインプロセッサの管理下にあるデータを操作する必要がある命令を例としたが、これに限定されることなく、ソフトウェアで処理した方が効率的な命令に遭遇した場合に、メインプロセッサ側に処理を依頼するようにしてもよい。この種の命令の例として、上述のメインプロセッサの管理下にあるデータの操作を伴う命令の他に、スタックフレーム（プログラム上で定義された一群のスタックデータの集合）の入れ替えが発生する命令、浮動小数点演算を実行する命令などがある。

## 【0074】

さらに、スタックアーキテクチャを採用するRISCプロセッサで処理可能な命令をコプロセッサで実行し、それ以外の複雑な処理を必要とする命令をメインプロセッサで実行するように、メインプロセッサおよびコプロセッサがそれぞれ処理対象とする命令を区分けしてもよい。

## 【0075】

図1に示した実施の形態1では、半導体集積回路のチップサイズ上の制限により、内蔵するスタックメモリ270の記憶容量が十分に確保できない場合がある。この場合、スタックメモリ270に書き込まれるデータのオーバーフロー及びアンダーフローを検出するための、ハードウェア手段もしくはソフトウェア手段を設ける必要がある。

## 【0076】

ハードウェア手段としては、例えば、スタックポインタ260Aが取り得る上限のアドレスと下限のアドレスとを値として保有するレジスタを設ければよい。このように構成すれば、スタックポインタの値が更新される度に、このレジスタの値と比較して、データのオーバーフロー及びアンダーフローを検出することができる。メインプロセッサ100への検出の通知は、実行不能な命令に遭遇した場合と同様、割り込みを発生するか、あるいは、制御レジスタの特定のビットを変化させることによって実行することができる。

## 【0077】

ソフトウェア手段としては、例えば、Java言語では、バイトコードと呼ばれる中間コードを格納するファイル中に、プログラムの個々の実行単位毎に、スタック上で使用されるデータの量すなわちスタックフレームの大きさが予め明示されているので、各プログラムの実行前にメインプロセッサ上のソフトウェアによりこれをチェックするようにしておけばよい。

## 【0078】

スタックのオーバーフローを検出した場合には、メインプロセッサ上のソフトウェアにより、スタックメモリ270中のデータを、データメモリ500に待避させる。また、アンダーフローを検出した場合には、データメモリ500に待避させたデータをスタックメモリ270に復元する。データの待避あるいは復元は、スタックメモリの全てのデータに対して一括して行うこともできるし、一部ずつ分割して行うこともできる。

## 【0079】

この実施の形態1によれば、容量の制限された高速なスタックメモリ270を

コプロセッサの内部に集積したことによって、スタック上のデータをオペランドとする演算を算術論理ユニット 2 8 0 により高速に実行することが可能となる。さらに、データメモリ 5 0 0 は、メインプロセッサ 1 0 0 とコプロセッサ 2 0 0 と異なる半導体集積回路上に実現すれば、オーバーフローの発生しない十分な記憶容量を確保することが容易に可能となる。

## 【 0 0 8 0 】

## ＜実施の形態 2＞

次に、この発明の実施の形態 2 を説明する。

J a v a 言語の命令体系では、スタックデータの先頭（上位）の 4 つのデータをレジスタ内に持つことにより、命令を効率的に実行し得るものとなっている。逆に言えば、先頭の 4 つのデータ以外のスタックデータについては、あえてレジスタに保持する必要はなく、外部のメモリに保持させておいても、命令の実行には支障がない。このように、スタックデータを外部のメモリに保持させることにより、スタックデータの容量を飛躍的に拡大することができる。

この実施の形態 2 では、上述のようにスタックデータの先頭の 4 つのデータのみをレジスタ内に保持し、J a v a 言語の処理に適したマイクロプロセッサシステムについて説明する。

## 【 0 0 8 1 】

図 7 に、この実施の形態 2 に係るプロセッサシステムの構成を示す。このプロセッサシステムは、前述の図 1 に示す実施の形態 1 にかかるプロセッサシステムの構成において、スタックメモリ 2 7 0 に代えてスタックトップレジスタ 2 7 0 A とスタックメモリ 2 7 0 B とを有するコプロセッサ 2 0 0 A を備える。また、この実施の形態では、命令キュー 2 3 0 は内部バス 2 0 1 にも接続され、メインプロセッサ 1 0 0 側から、命令キュー 2 3 0 の内容を直接的に読み出すことが可能なようになっている。

## 【 0 0 8 2 】

ここで、スタックトップレジスタ 2 7 0 A は、算術論理ユニット 2 8 0 に接続され、スタックデータのうち、先頭側の 4 つのデータ（所定数のデータ）を保持するものである。このスタックトップレジスタ 2 7 0 A は、リードとライトが同

時に且つ高速に行い得るように構成されたものである。また、スタックメモリ 270B は、上述の実施の形態 1 にかかるスタックメモリ 270 に相当するものであって、外部バス 600 とスタックトップレジスタ 270A とに接続されて、コプロセッサ 200A の外部に設けられ、スタックデータの上位の 4 つのデータの下位側のデータを保持する。この構成の場合、スタックポインタ 260A の SP 値は、スタックメモリ 270B の先頭アドレスを示す。

#### 【0083】

ここで、図 8 を参照して、スタックトップレジスタとスタックメモリの一般的な動作概念を補足説明しておく。スタックトップレジスタおよびスタックメモリは、上述のスタックメモリ 270 と同様に LIFO 方式でデータの書き込みと読み出しが行われるものであって、同図 (a) に示す例では、スタックデータ D11 ~ D15 のうち、先頭の 2 つのデータ D14, D15 がスタックトップレジスタに保持される。

#### 【0084】

以下、2 項演算を行う場合を例として、スタックトップレジスタおよびスタックメモリの内容の変化の様子を示す。演算に先だって、図 8 (a) に示すように、アドレス「0x1004」~「0x1014」には、データ D11 ~ D15 が順次積み重ねられるように収納される。この場合、スタックメモリに最後に収納されたデータ D13 のアドレス「0x100C」が先頭アドレスとなり、スタックポインタの値とされる。この状態から、二項演算（例えば二項の加算演算）が実行されると、スタックトップレジスタ内のデータ D14 とデータ D15 とに対して演算がなされる。

#### 【0085】

具体的には、この演算が「加算」であれば、データ D14 とデータ D15 とが加算され、その演算結果としてデータ D16 が得られる。この時点で、データ D14, D15 は不要となる。そこで、図 8 (b) に示すように、演算結果として算出されたデータ D16 は、空いたアドレスの下位側、すなわち後続のデータ D13 の次のアドレス「0x1014」に保存される。このとき先頭側のデータ D16 およびその後続のデータ D13 の 2 つのデータは、スタックトップレジスタ

側に保持される。このように、スタックトップレジスタには、演算を処理する過程で、スタックデータの先頭側の所定数のデータが一時的に保持される。

#### 【0086】

なお、図8による補足説明では、スタックトップレジスタは、先頭側の2つのデータを保持するものとしたが、図7に示すこの実施の形態2に係る構成では、スタックデータのうち、先頭の4つのデータをスタックトップレジスタ270Aに保持するものである。ただし、この発明は、これに限定されるものではなく、スタックトップレジスタのデータ数は、必要に応じた任意の数に設定される。

#### 【0087】

以下、図9に示すフローチャートに沿って、スタックフレーム（プログラム上で定義された一群のスタックデータの集合）を入れ替える場合を含む動作を例として、この実施の形態2の動作を説明する。

最初に、このマイクロプロセッサシステムがリセットされる。これにより、各部の動作が初期化され、このリセット後にメインプロセッサ100が、動作を開始し、コプロセッサ200Aが停止状態に置かれる。

#### 【0088】

ステップST101：この状態から、メインプロセッサ100は、ソフトウェア上での操作により、スタックの先頭アドレスをコプロセッサ200A内のスタックポインタ260Aに設定し、コプロセッサ200Aが実行すべきプログラムの先頭アドレスをプログラムカウンタ260Bに設定する。これにより、コプロセッサ200Aのスタックポインタ260Aおよびプログラムカウンタ260Bに初期値がそれぞれ設定される。

#### 【0089】

ステップST102：続いて、メインプロセッサ100は、ソフトウェア上での操作により、コプロセッサ200A内の制御レジスタ210に、「動作開始」を指示するための制御データ（例えばデータ「0」）を設定し、コプロセッサ200Aの動作を開始させる。

#### 【0090】

ステップST103：続いて、動作を開始したコプロセッサ200Aは、ハー

ドウェア上での操作により、プログラムカウンタ 260B に設定されたアドレスを参照して、第 2 の命令セットの中間コードが展開されたプログラムメモリ 400 をアクセスし、このアドレスから自発的に第 2 の命令をフェッチする。フェッチされた命令は命令キュー 230 に保持される。そして、コプロセッサ 200A は、フェッチされた命令を命令デコーダ 240 にて解読（デコード）する。

ステップ ST104：続いて、コプロセッサ 200A は、フェッチされた命令がスタックフレームを変更する命令か否かを判断する。

#### 【0091】

ここで、スタックトップレジスタが先頭側の 2 つのデータを保持するものである場合を例とし、図 10 を参照して、スタックフレームの変更の概念を補足説明しておく。この例では、フレーム変更前において、スタックメモリ 270B には、引数データ D21、ローカル変数 D22、コンテキスト D23、演算データ D24、および引数データ D25 の一部（引数データ d25-n）が保持されている。一方、スタックトップレジスタ 270A には、引数データ D25 の残り（引数データ d25-1, d25-2）が保持されている。これら先頭側のデータ D21～D25 が、1 つのスタックフレームを構成する。

#### 【0092】

フレーム変更を行うと、変更前のフレームに新たなデータが積み重ねられ、この先頭側のデータ群が新たなスタックフレームを構成する。この図に示す例では、それまでのデータ D25 に積み重ねるようにして、ローカル変数 D26、コンテキスト D27 が保持され、その先頭側の引数データ D25～コンテキスト D27 が新たなスタックフレームを構成する。

#### 【0093】

フレーム変更後では、スタックメモリ 270B には、引数データ D21～引数データ D25 に加え、新たなデータとして、ローカル変数 D26、およびコンテキスト D27 の一部（コンテキスト d27-n）が保持される。一方、スタックトップレジスタ 270A には、コンテキスト D27 の残り（コンテキスト d27-1, d27-2）が保持される。この一連のスタックフレームの変更処理は、メインプロセッサ 100 のソフトウェア上で行われる。

## 【0094】

上述のように、スタックフレームを変更する場合、スタックポインタの値が大きく変わる。この処理を行う場合、スタックトップレジスタ270Aとスタックメモリ270Bとにわたってスタックデータを操作する必要が生じ、メインプロセッサ100での処理が複雑化する。

そこで、以下のステップで説明するように、スタックフレームの変更処理を実行する前に、スタックトップレジスタ270Aの全ての内容を、予めスタックメモリ270Bに移しておく。これにより、スタックフレームを変更するための処理では、スタックメモリ270Bのみを参照すればよく、この処理の負荷を軽減することができる。

## 【0095】

ここで、再び説明を図9に示すフローチャートに戻す。

ステップST105：上述のステップST104において、フェッチされた命令がスタックフレームを変更する命令であると判断された場合（ステップST104：YES）、コプロセッサ200Aは、スタックトップレジスタ270Aに保持されている全ての内容をスタックメモリ270Bに書き込む。そして、コプロセッサ200Aは、この書き込みによるスタックメモリ270Aのデータの増分に応じて、スタックポインタ260AのSP値を更新する。

## 【0096】

ステップST106：続いて、コプロセッサ200Aは、フェッチされた命令が専用の割り込みハンドラを持つか否かを判断して、割り込み要求を発生し、メインプロセッサ側において割り込みハンドラの特定制が行われる。このステップST106での処理は、上述の実施の形態1にかかるステップST20～ST23、ST30～ST34（図6参照）による処理と同様に行われる。

## 【0097】

ステップST107：続いて、スタックフレームを変更するための演算処理を行う。具体的には、メインプロセッサ100は、全てのスタックデータが保持されたスタックメモリ270Bまたはデータメモリ500から処理に必要なデータを読み出し、新たなスタックフレームを準備する。続いて、コプロセッサ200



A内のスタックポインタ260AのSP値（アドレス）およびプログラムカウンタ260BのPC値（アドレス）を読み出して保存する。続いて、このスタックポインタ260Aおよびプログラムカウンタ260Bに新たな値（アドレス）を書き込む。そして、スタックメモリ270Bまたはデータメモリ500に演算結果を書き戻す。続いて、コプロセッサ200Aの制御レジスタ210に「操作開始」指示する制御データを設定し、コプロセッサ200Aに動作を再開させる。

## 【0098】

ステップST108：動作を再開したコプロセッサ200Aは、メインプロセッサ100の演算結果が収納されたスタックメモリ270Bから、スタックトップレジスタ270Aのデータ数分のデータを読み込み、これをスタックトップレジスタ270Aに保持する。そして、この読み込みによるスタックメモリ270Bのデータの減少分に応じて、スタックポインタ260AのSP値を更新する。

ステップST109：この後、コプロセッサ200Aは、プログラムカウンタ260BのPC値を、「0」を加算したものとして更新して、処理を上述のステップST103に戻し、次の命令を処理対象とする。

## 【0099】

ステップST110：次に、上述のステップST104において、スタックフレームを変更する命令でない場合（ステップST104：NO）、コプロセッサ200Aは、フェッチされた命令が、スタックを増加させる命令か否かを判断する。つまり、スタックメモリにスタックされるデータが発生する場合には、スタックメモリに空き領域を確保する必要があるが生じる。

## 【0100】

このステップでの判断は、換言すれば、演算に先だって、この空き領域を予め確保するか否かを判断するものである。なお、スタックが減少する場合には、演算を実行する前に、スタックポインタを減少させることは許されない。なぜなら、スタックポインタを減少させることにより、演算に必要なデータが無効とされる場合が発生し得るからである。

## 【0101】

ステップST111：ここで、スタックを増加させる命令である場合（ステッ

ブST110: YES)、コプロセッサ200Aは、必要分のスタックトップレジスタ270Aの内容をスタックメモリ270Bに書き込み、スタックポインタ260Aの値を更新する。また、スタックを増加させる命令でない場合(ステップST110: NO)、このステップST111をパスする。

#### 【0102】

ステップST112: 続いて、コプロセッサ200Aは、フェッチされた命令が単独で処理可能な命令か否かを判断し、この判断結果に応じて、コプロセッサ200Aのハードウェア上での処理、またはメインプロセッサ100のソフトウェア上での処理が行われ、この命令に対する処理が適応的に行われる。このステップST112での処理は、具体的には、上述の実施の形態1にかかるステップST15~18, ST20~ST26, ST30~ST34(図5参照)による処理と同様である。

【0103】 この実施の形態2では、メインプロセッサ100は、コプロセッサ200Aの命令キュー230を直接的にアクセスできるので、メインプロセッサ100がコプロセッサから依頼された命令を処理する場合、コプロセッサ200Aの命令キュー230を参照して、処理すべき命令の種類を特定し、この命令に対応する割り込みハンドラを特定することができる。これにより、専用の割り込みベクタが割り付けられていない命令であっても、即座に命令の種類が特定される。

#### 【0104】

ステップST113: 続いて、コプロセッサ200Aは、フェッチされた命令がスタックを減少させる命令であるか否かを判断する。

ステップST115: ここで、スタックを減少させる命令である場合(ステップST113: YES)、コプロセッサ200Aは、必要分のスタックトップレジスタの内容をスタックメモリから読み込み、スタックポインタ260Aの値を更新する。また、スタックを減少させる命令でない場合(ステップST113: NO)、このステップST114をパスする。

この後、処理は上述のステップST109に移行し、プログラムカウンタ260BのPC値が更新される。そして、上述のステップST103に戻され、次の

命令が処理の対象とされる。

【0105】

以上のように、この実施の形態2によれば、コプロセッサ200Aにおいて、命令デコーダにより解読された命令がスタックポインタを非連続的な領域に変更する場合（例えば上述のスタックフレームを変更するような場合）、メインプロセッサ100に通知する前に、スタックトップレジスタ270Aの全てのスタックデータをスタックメモリに書き出し、メインプロセッサ100での処理が終了して動作が復帰した後に、スタックメモリ270Bからスタックトップレジスタ270Aに先頭側のデータを読み出す。

【0106】

これにより、メインプロセッサ100は、割り込みハンドラ内にて、スタックデータを単一のメモリ空間上で取り扱うことができ、スタックトップレジスタ270Aのデータを扱う場合の処理が簡略化される。したがって、スタックトップレジスタ270Aを有するアーキテクチャと親和性が高いJavaなどのプログラム言語の命令を効率的に実行することが可能となる。

【0107】

また、この実施の形態2によれば、コプロセッサ200Aは、スタックメモリ270Bを外部に持つので、スタックメモリ270Bの記憶容量を大きくすることができる。可能であれば、このスタックメモリ270Bをコプロセッサ200Aの内部に持つようにしてもよい。

【0108】

<実施の形態3>

次に、この発明の実施の形態3を説明する。

図11に、この実施の形態3にかかるマイクロプロセッサシステムの構成を示す。

このマイクロプロセッサシステムは、上述の図7に示す実施の形態2に係る構成において、スタックトップレジスタ270Aとスタックメモリ270Bとの間に、キャッシュメモリ270Cを有するコプロセッサ200Bを備える。このキャッシュメモリ270Cは、スタックメモリ270Bに保持されたデータの一部

をキャッシュするものである。

なお、この実施の形態 3 では、スタックメモリ 270B と外部バス 600 との間の接続は切り離されている。

【0109】

このように、キャッシュメモリ 270C を備えることにより、見かけ上、外部のスタックメモリ 270B を高速にアクセスすることが可能となり、処理の高速化が図れる。この場合、スタックメモリ 270B とキャッシュメモリ 270C との間のデータの整合を保つ必要上、スタックメモリ 270B は、必ずキャッシュメモリ 270C を介してアクセスされる。このため、スタックメモリ 270B は外部バス 600 からの直接アクセスが不能なように、この外部バス 600 との接続が切り離されている。

すなわち、メインプロセッサ 100 からスタックメモリ 270B に格納されるべきデータにアクセスするためには、外部バス 600 とコプロセッサ 200 の内部バス 201 とを経由し、さらにキャッシュメモリ 270C を通してアクセスしなければならない。

【0110】

この実施の形態 3 によれば、キャッシュメモリ 270C を介してスタックメモリ 270B がアクセスされるので、スタックメモリ 270B を高速にアクセスすることができ、仮にスタックメモリ 270B が高速にアクセスできないものであっても、見かけ上の処理を高速化することができる。

また、この実施の形態 3 によれば、コプロセッサ 200B は、スタックメモリ 270B を外部に持つので、スタックメモリ 270B の記憶容量を大きくすることができる。

【0111】

さらに、この実施の形態 3 によれば、キャッシュメモリ 270C 自体にキャッシュの対象となるデータを管理する機能が付加されているので、外部に接続されたスタックメモリ 270B の記憶容量を十分大きくしてオーバーフローに対する対策を講じる必要がない。すなわち、実施の形態 1 では、ハードウェア手段またはソフトウェア手段により、オーバーフローあるいはアンダーフローの検出を行

っているが、この実施の形態3では、オーバーフロー検出手段、アンダーフロー検出手段が不要であり、また、オーバーフローまたはアンダーフローの検出に伴うメモリ入れ換え作業が不要となるので、チップサイズの縮小および制御用ソフトウェアのプログラムコードサイズの縮小が実現可能となる。

#### 【0112】

#### ＜実施の形態4＞

次に、この発明の実施の形態4を説明する。

図12に、この実施の形態4にかかるマイクロプロセッサシステムの構成を示す。同図に示すように、このプロセッサシステムは、1つのメインプロセッサ100に対して、複数のコプロセッサ200X, 200Y, 200Zが設けられている。コプロセッサ200X, 200Y, 200Zは、前述の実施の形態1にかかるスタックメモリ270に相当する専用のスタックメモリ270XM, 270YM, 270ZMをそれぞれ持つ。

#### 【0113】

これらコプロセッサ200X, 200Y, 200Zは、外部バス600を介してメインプロセッサ100に接続されると共に、外部バス600Bを介してプログラムメモリ400に接続される。また、プログラムメモリ400およびデータメモリ500は外部バス600を介してメインプロセッサ100に接続される。各コプロセッサからの割り込み要求S290Hはメインプロセッサ100に直接入力される。

#### 【0114】

この実施の形態4では、各コプロセッサのスタックメモリ270XM, 270YM, 270ZMには、それぞれのコプロセッサに固有のローカルなデータ（例えばローカル変数、ローカルな演算データ、ローカルなワーキングデータ）が格納される。このローカルなデータとしては、例えばJava言語の場合、プログラムに記述されたタスクの単位である「スレッド」に相当するデータの集合が格納される。

#### 【0115】

この実施の形態4では、メインプロセッサ100が処理すべき命令とコプロセ

ッサ200X, 200Y, 200Zが処理すべき命令は、データメモリ500上に設けられたヒープ領域（オブジェクトが書き込まれる領域）を操作する必要があるか否かにより区別される。すなわち、ヒープ領域を操作する必要がある命令については、メインプロセッサ100で処理を行い、ヒープ領域を操作する必要のない命令については、コプロセッサ200X, 200Y, 200Zで処理を行う。

【0116】

また、メインプロセッサ100は、データメモリ500上のヒープ領域に作成されたオブジェクトへのアクセスが複数のコプロセッサのスレッドで発生したときに、各スレッド間の排他制御を行う。

具体的には、各コプロセッサは、固有のローカルデータが格納されたスタックメモリ270XM, 270YM, 270ZMをアクセスして実行可能な命令のみを処理の対象とする。

【0117】

また、メインプロセッサ100は、各コプロセッサ間の調整を伴う命令を処理の対象とする。例えば、Java言語の「`invokevirtual`」命令は、ヒープ領域中のオブジェクトに対し複数の処理が記述されたプログラム中の特定の処理を選択的に実行する。このとき、実行の対象となるオブジェクトが他のスレッドによって排他的に使用されている場合には、その使用が終了するまでプログラムの実行を待たなければならない。メインプロセッサ100は、実行中のスレッドの終了を待ち、次の要求のあったコプロセッサにアクセスして、プログラムカウンタに実行すべき処理が記述されたアドレスを設定後、該当するコプロセッサの制御レジスタを動作開始に設定する。

【0118】

ただし、Java言語の「ガベージコレクション」すなわち、ヒープ領域上のオブジェクトを操作して、不要となったデータを廃棄する処理については、コプロセッサによる割り込みとは無関係に、メインプロセッサ100が実行する。

なお、メインプロセッサ100は、「ガベージコレクション」の処理中に各コプロセッサからの割り込みを受け付けないように、予めこの割り込みをマスクす

ることにより、ガベージコレクションに伴う格納場所の移動中のオブジェクトの書き換えを防止する。

#### 【0119】

この実施の形態4によれば、複数のコプロセッサ200X, 200Y, 200Zにより、マルチタスクの処理が可能となり、Java言語などのようにマルチスレッドの概念を有するプログラムを高速に実行することが可能となる。

また、各コプロセッサでの処理と並行して、メインプロセッサ100がガベージコレクションなどの処理を行うことができ、したがって全体の処理時間を有効に短縮することができる。

#### 【0120】

さらに、メインプロセッサ100は、各コプロセッサからの割り込み処理をマスクすることにより、オブジェクト操作要求を容易に制御することができ、このため、ガベージコレクションを容易に且つリアルタイムに行うことができる。

さらにまた、データメモリ500上のヒープ領域に作成されたオブジェクトへのアクセスが複数のコプロセッサのスレッドで発生したときに、各スレッド間の排他制御をメインプロセッサ上で一括処理するので、コプロセッサ側に排他制御のための手段やプログラムを用意しなくてもよく、また、メインプロセッサ上の排他制御処理プログラムが簡略化できる。

#### 【0121】

以上、この発明の実施の形態1ないし4を説明したが、この発明は、これらの実施の形態に限られるものではなく、この発明の要旨を逸脱しない範囲の設計変更等があっても本発明に含まれる。例えば、上述の各実施の形態では、コプロセッサ側からメインプロセッサ側に割り込みを発生させることにより、通知するものとしたが、これに限定されることなく、たとえばコプロセッサ側に状態レジスタを設け、この状態レジスタに割り込み要求を通知するためのデータを書き込んでおき、この状態レジスタをメインプロセッサが定期的にアクセスすること（ポーリング）により、コプロセッサが処理不能な特定の命令に遭遇したことをメインプロセッサ側で把握するものとしてもよい。これにより、結果的に、メインプロセッサ側に「通知」が行われ、メインプロセッサでその処理が実行される。

## 【0122】

なお、この発明は、見方を変えれば、コプロセッサが遭遇した命令の内容に応じて、命令の処理形態を選択するものとして把握することができる。例えば、コプロセッサが持つスタックデータのみで処理が可能な命令についてはコプロセッサ自身で処理を行い、またデータメモリ上のヒープ領域を操作する必要がある命令についてはメインプロセッサで処理を行い、さらにスタックフレームを変更する必要がある命令についてもメインプロセッサで処理を行うように、処理の主体を選択する。

## 【0123】

## 【発明の効果】

以上説明したように、この発明によれば、プログラムに記述された命令を実行するマイクロプロセッサシステムであって、第1の命令セットをハードウェア上で実行すると共に第2の命令セットをソフトウェア上で実行するメインプロセッサと、前記メインプロセッサの管理下で動作して前記第2の命令セットをハードウェア上で実行するコプロセッサとを備えたので、回路規模の増大を抑えつつ、特定の命令セット（第2の命令セット）を高速に実行することが可能となる。

## 【0124】

また、前記コプロセッサが、前記第2の命令セットのうち、前記メインプロセッサの管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサに通知を発行して当該命令の実行を依頼するようにしたので、メインプロセッサのソフトウェアが有効に活用されると共にコプロセッサのハードウェアの規模を必要最小限に抑えることが可能となり、したがって全体としてハードウェアの規模を有効に抑えることが可能となる。

## 【0125】

さらに、前記コプロセッサが、前記第2の命令セットのうち、実行頻度の高い所定の複数の命令に対して予め割り付けられた専用割り込みベクタを用いて前記通知を発行するようにしたので、メインプロセッサにおいて、コプロセッサが遭遇した命令が即座に特定される。したがって、メインプロセッサ側において、コプロセッサからの依頼により処理すべき命令を識別するための処理を省くことが



可能となる。

【0126】

さらにまた、前記コプロセッサが、前記第2の命令セットに属する命令を実行する過程で発生するデータを保持するためのスタックメモリと、該スタックメモリ内の最新データのアドレスを保持するスタックポインタとを有し、前記特定の命令を実行する過程で発生する処理のうち、前記スタックメモリのスタックポインタを更新するための処理を実行するハードウェア資源を備えたので、メインプロセッサが依頼を受けてこの命令を処理する過程において、スタックポインタを更新するための処理を省くことができる。したがって、メインプロセッサのソフトウェア上での処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を短縮することができる。

【0127】

さらにまた、前記コプロセッサが、現在処理中の前記第2の命令セットに属する命令のアドレスを保持するためのプログラムカウンタを有し、前記特定の命令を実行する過程で発生する処理のうち、前記プログラムカウンタを更新するための処理を実行するハードウェア資源を備えたので、メインプロセッサに処理を依頼する場合であっても、プログラムカウンタを更新するための処理については、コプロセッサのハードウェア上で行われ、メインプロセッサがプログラムカウンタを更新するためのソフトウェア上の処理を省くことができる。したがって、メインプロセッサのソフトウェア上の処理が簡略化されると共に、コプロセッサのハードウェアが有効に活用され、メインプロセッサでのソフトウェアによる処理時間を有効に短縮することができる。

【0128】

さらにまた、前記コプロセッサが、前記通知を行う必要が発生したことを表す情報を保持するための状態レジスタを有し、前記メインプロセッサが、前記状態レジスタを定期的にアクセスして、該状態レジスタの内容から前記コプロセッサが前記特定の命令に遭遇したことを把握して、前記特定の命令を実行するようにしたので、メインプロセッサにより、コプロセッサが特定の命令に遭遇したか否

かが定期的に把握され、メインプロセッサに対して割り込みベクタを用いて割り込み処理を行う必要がなくなる。したがって、コプロセッサからメインプロセッサへの通知を簡略化することが可能となる。

## 【 0 1 2 9 】

さらにまた、前記メインプロセッサが、前記コプロセッサからの前記専用割り込みベクタをエンコードして、処理すべき前記特定の命令に対応する割り込みハンドラを特定する割り込み要求受付回路を備えたので、メインプロセッサにおいて、コプロセッサから入力する専用割り込みベクタに基づき直接的に割り込みハンドラが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウェア上の処理が不要となり、メインプロセッサでの処理時間を短縮することが可能となる。

## 【 0 1 3 0 】

さらにまた、前記メインプロセッサが、前記コプロセッサの命令キューを参照して、処理すべき前記特定の命令に対応する割り込みハンドラを特定するようにしたので、メインプロセッサにおいて、コプロセッサの命令キューを参照してコプロセッサが遭遇した命令が把握され、直接的に処理ルーチンが特定される。したがって、メインプロセッサ側で起動すべき割り込みハンドラを特定するためのソフトウェア上の処理が簡略化され、メインプロセッサでの処理時間を短縮することが可能となる。

## 【 0 1 3 1 】

さらにまた、前記コプロセッサが、スタックアーキテクチャを有するようにしたので、コプロセッサにおいて、サブルーチンの頻繁な使用を伴う命令や、ゼロオペランドのアドレッシングを伴う命令の処理が効率的に行われる。したがって J a v a などのスタックアーキテクチャに基づくインタプリタ言語の命令を効率的に実行することが可能となる。

## 【 0 1 3 2 】

さらにまた、スタックアーキテクチャを採用する前記コプロセッサが、スタックデータのうち、先頭側の所定数のデータを保持するスタックトップレジスタと、外部に設けられたスタックメモリと前記スタックトップレジスタとの間に設け

られ、前記スタックメモリに保持されたデータの一部をキャッシュするキャッシュメモリと、を含むようにしたので、スタックデータのうち、先頭側の所定数のデータをアクセスする頻度の高いプログラム言語の命令を効率的に実行することができると共に、スタックメモリの容量を適応的に拡大することが可能となる。

【0133】

さらにまた、プログラムに記述された複数の処理内容に対応づけて、複数の前記コプロセッサを備えたので、プログラム中の複数の処理を並列処理することが可能となり、処理を高速化することが可能となる。

【図面の簡単な説明】

【図1】 この発明の実施の形態1にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【図2】 この発明の実施の形態1にかかるスタックメモリの動作概念を説明するための説明図である。

【図3】 この発明の実施の形態1にかかる命令デコーダの詳細とその周辺の構成を示すブロック図である。

【図4】 この発明の実施の形態1にかかる割り込み要求発生回路と割り込み要求受付回路の詳細な構成を示すブロック図である。

【図5】 この発明の実施の形態1にかかるマイクロプロセッサシステムの動作（命令がコプロセッサで処理可能な場合）の流れを示すフローチャートである。

【図6】 この発明の実施の形態1にかかるマイクロプロセッサシステムの動作（命令がコプロセッサで処理不能な場合）の流れを示すフローチャートである。

【図7】 この発明の実施の形態2にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【図8】 この発明の実施の形態2にかかるスタックトップレジスタの動作概念を説明するための説明図である。

【図9】 この発明の実施の形態2にかかるマイクロプロセッサシステムの動作の流れを示すフローチャートである。

【図 10】 この発明の実施の形態 2 にかかるスタックトップレジスタの動作概念（スタックフレームを変更する場合）を説明するための説明図である。

【図 11】 この発明の実施の形態 3 にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【図 12】 この発明の実施の形態 4 にかかるマイクロプロセッサシステムの構成を示すブロック図である。

【符号の説明】

- 100：メインプロセッサ
- 110：命令フェッチ回路
- 120：命令キュー
- 130：命令デコーダ
- 140：加算器
- 150：プログラムカウンタ
- 160：汎用レジスタ
- 170：算術論理ユニット
- 180：データアクセス回路
- 190：割り込み要求受付回路
- 191：フリップフロップ
- 192：割り込みハンドラアドレス発生回路
- 193：割り込みアドレスレジスタ
- 200：コプロセッサ
- 200X～200Z：コプロセッサ
- 201：内部バス
- 210：制御レジスタ
- 220：命令フェッチ回路
- 221：整列回路
- 230：命令キュー
- 240：命令デコーダ
- 240A：デコーダ

240B : 処理手順制御信号発生回路

240C : SP用増減値発生回路

240D : PC用増減値発生回路

250A : 加減算器

250B : 加算器

260A : スタックポインタ

260B : プログラムカウンタ

270 : スタックメモリ

270A : スタックトップレジスタ

270B : スタックメモリ

270C : キャッシュメモリ

280 : 算術論理ユニット

290 : 割り込み要求発生回路

291 : 割り込み処理用デコーダ

292 : フリップフロップ

300 : プログラムメモリ (メインプロセッサ用)

400 : プログラムメモリ (コプロセッサ用)

500 : データメモリ

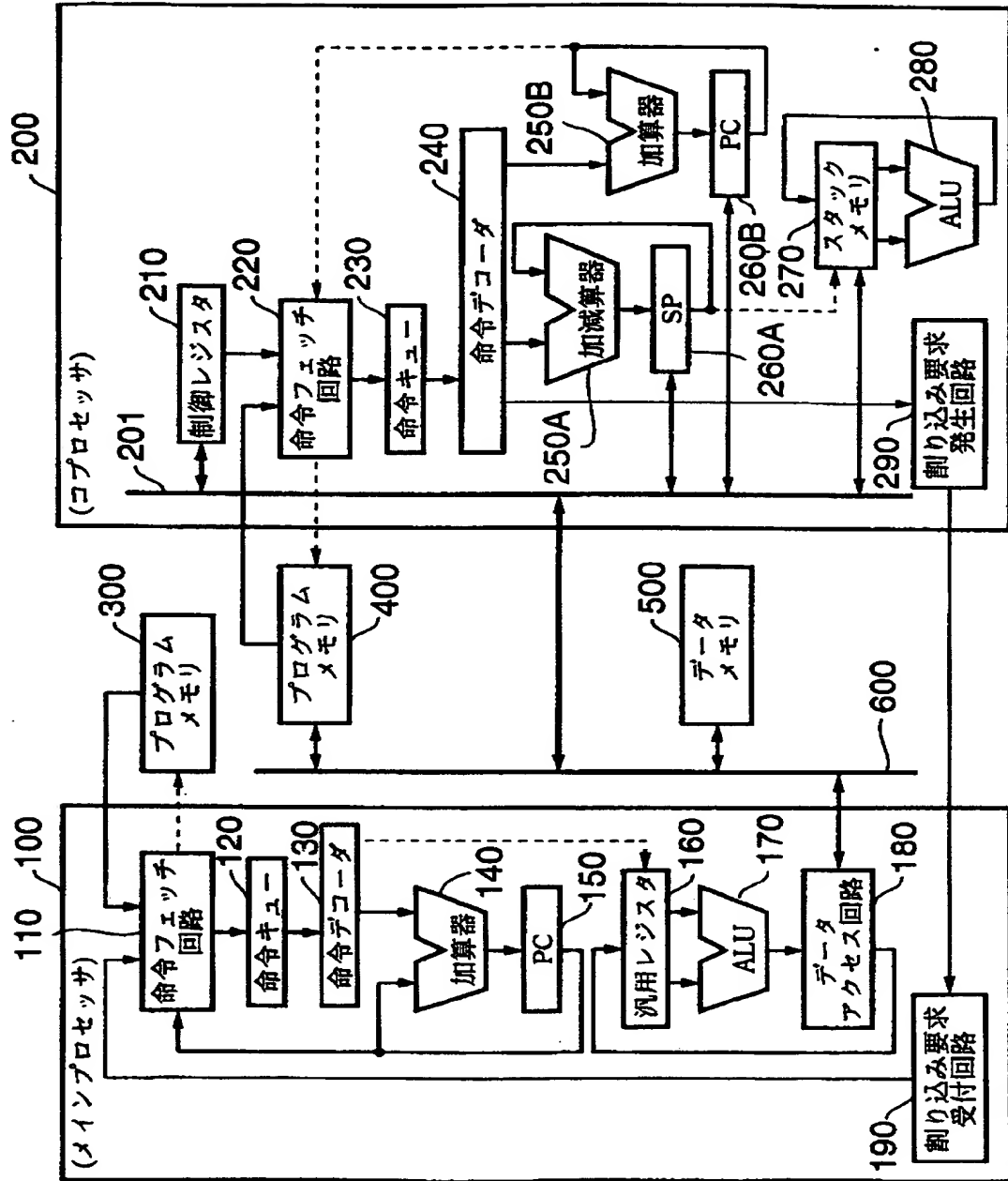
600, 600B : 外部バス

ST10~ST26, ST30~ST34, ST101~ST114 : ステッ

プ

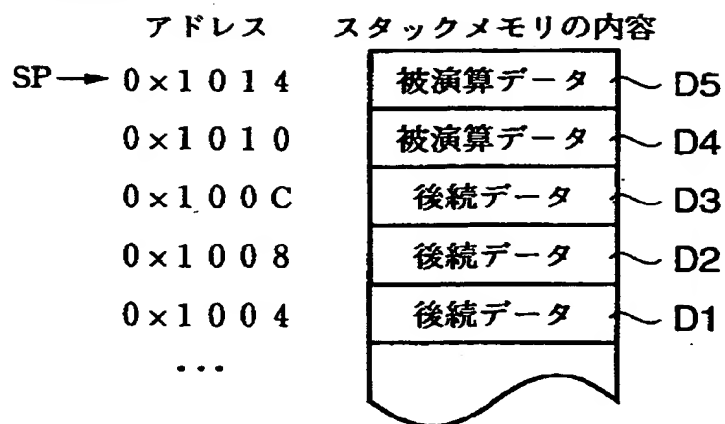
【書類名】 図面

【図 1】



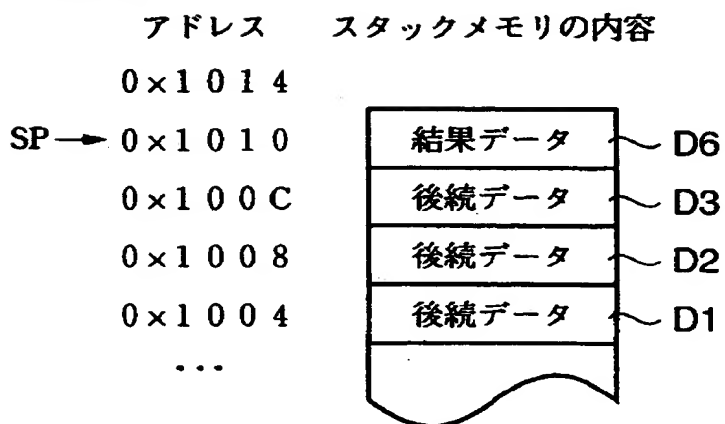
【図 2】

<演算前>



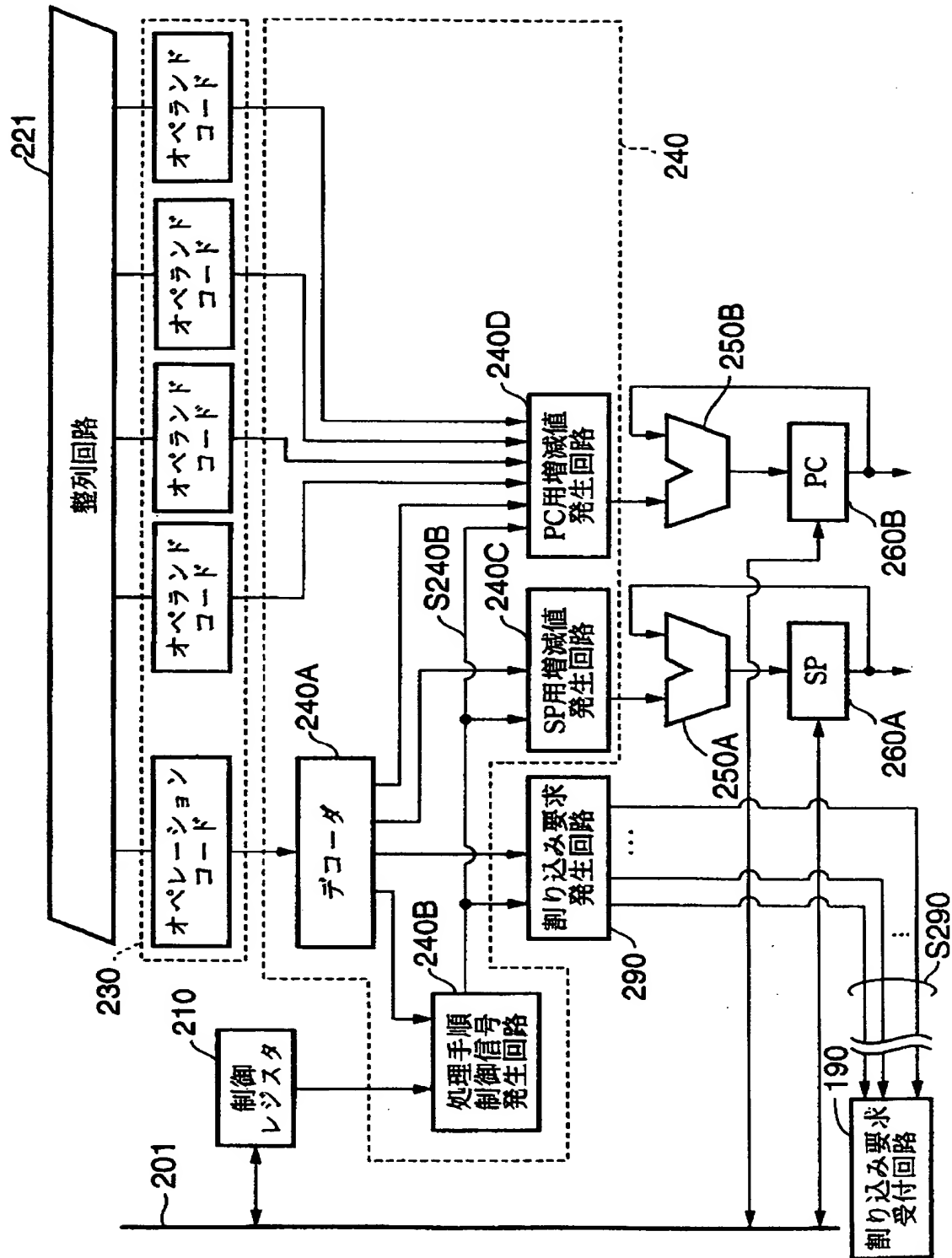
(a)

<演算後>



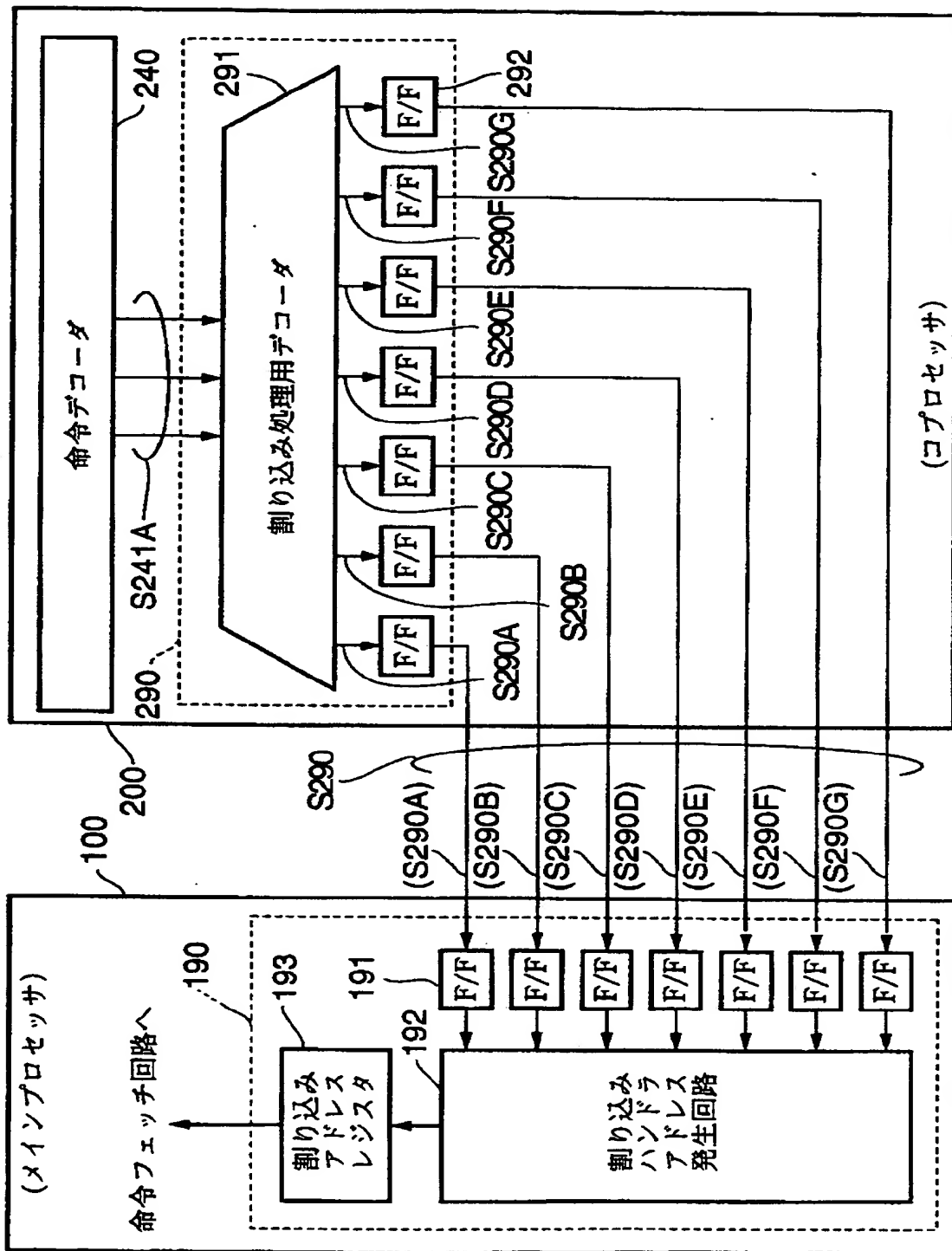
(b)

【図 3】

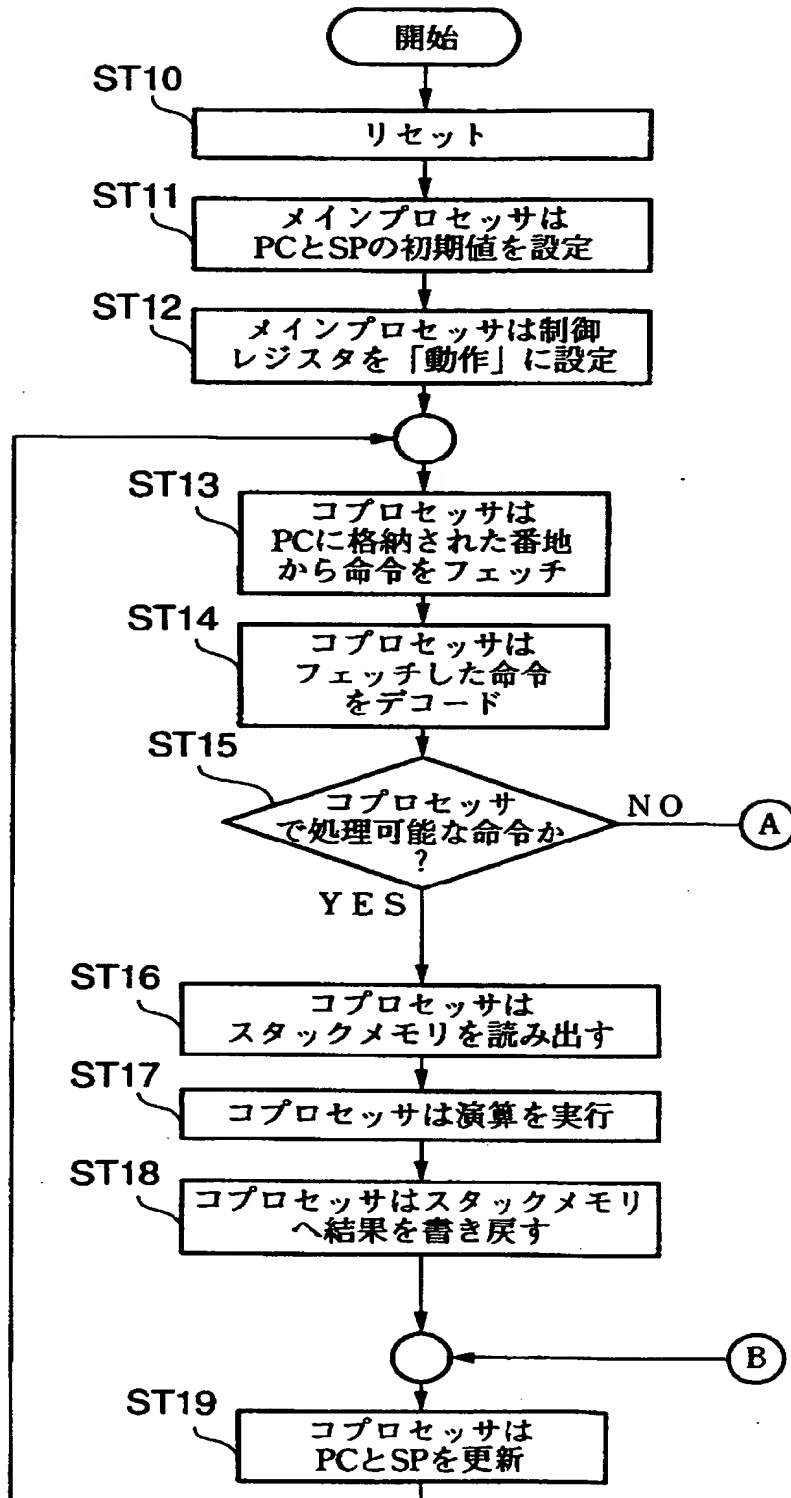




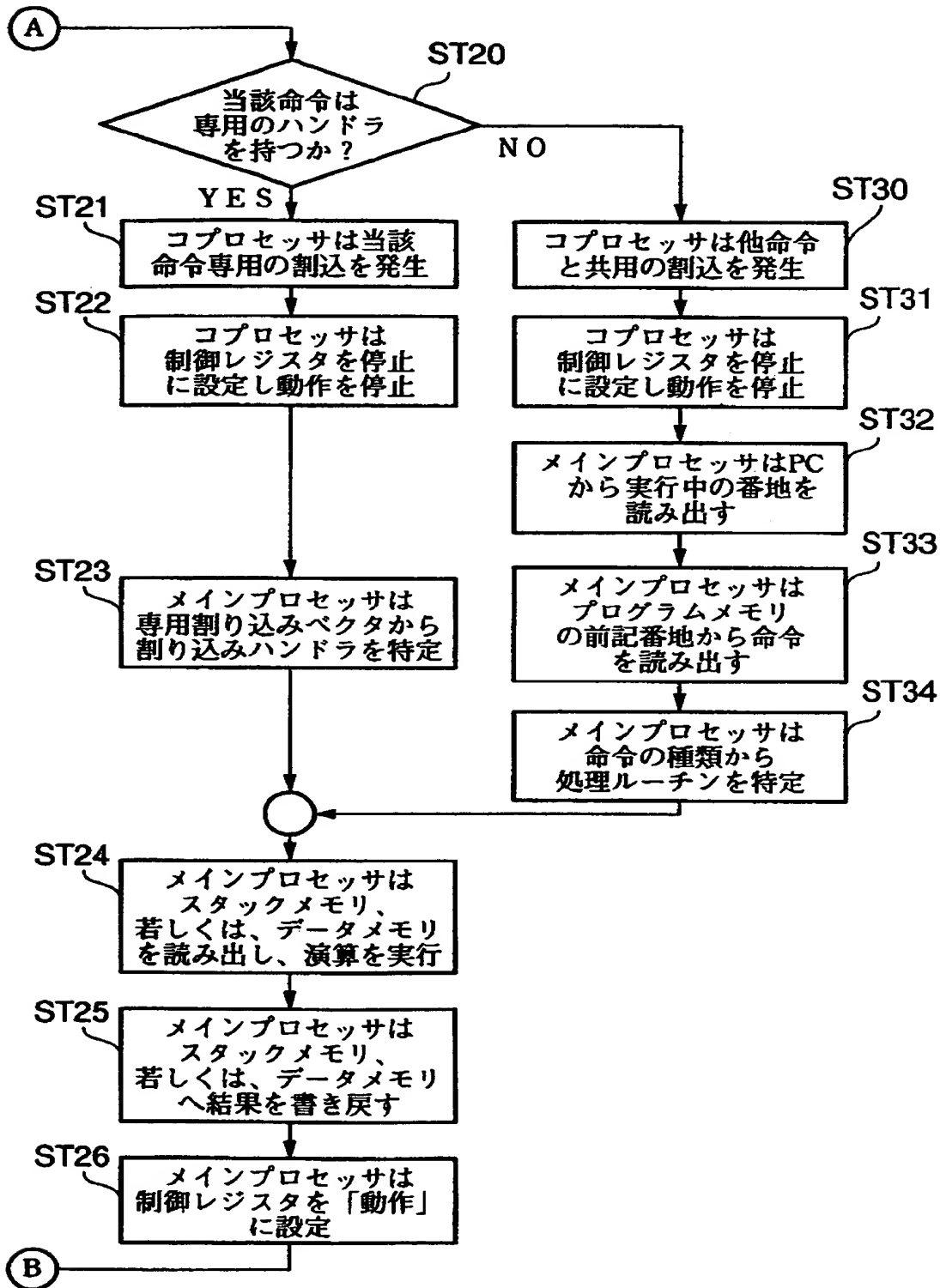
【図 4】



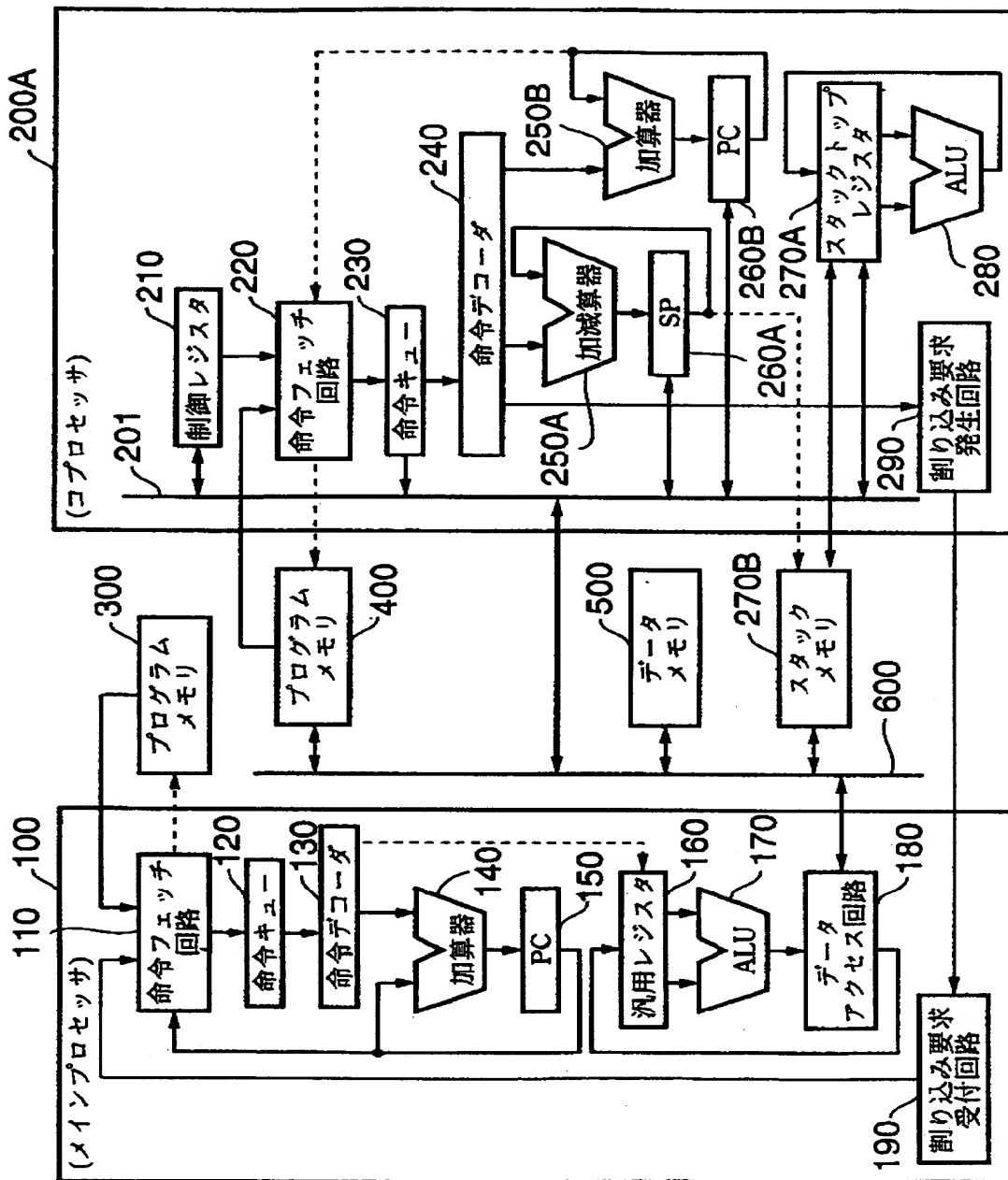
【図 5】



【図 6】

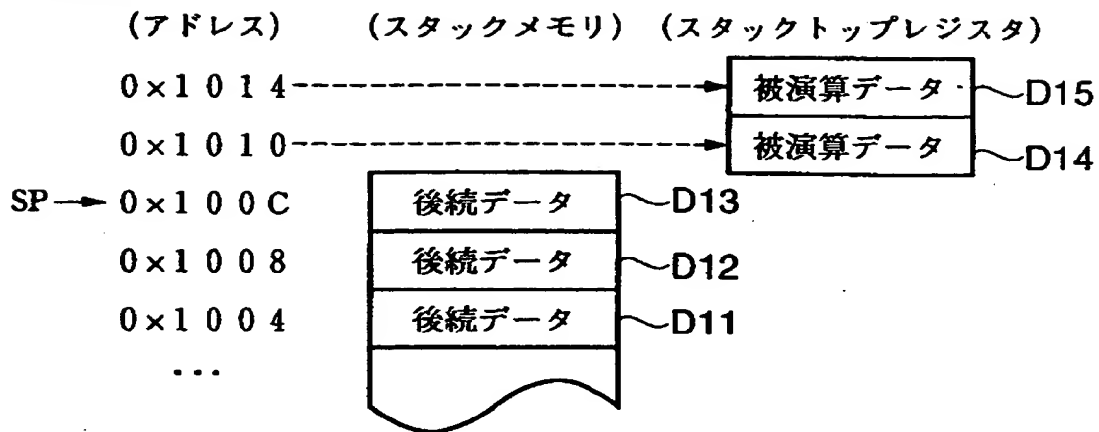


【図 7】



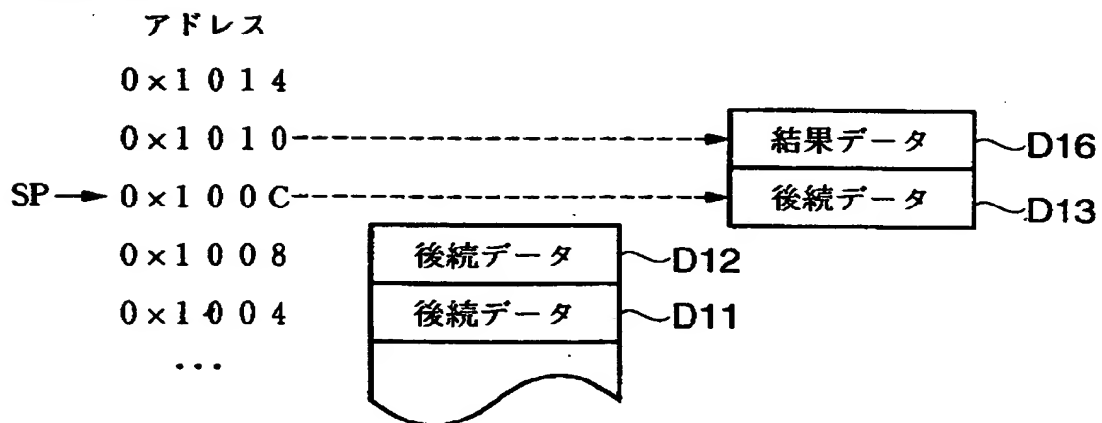
【図 8】

<演算前>



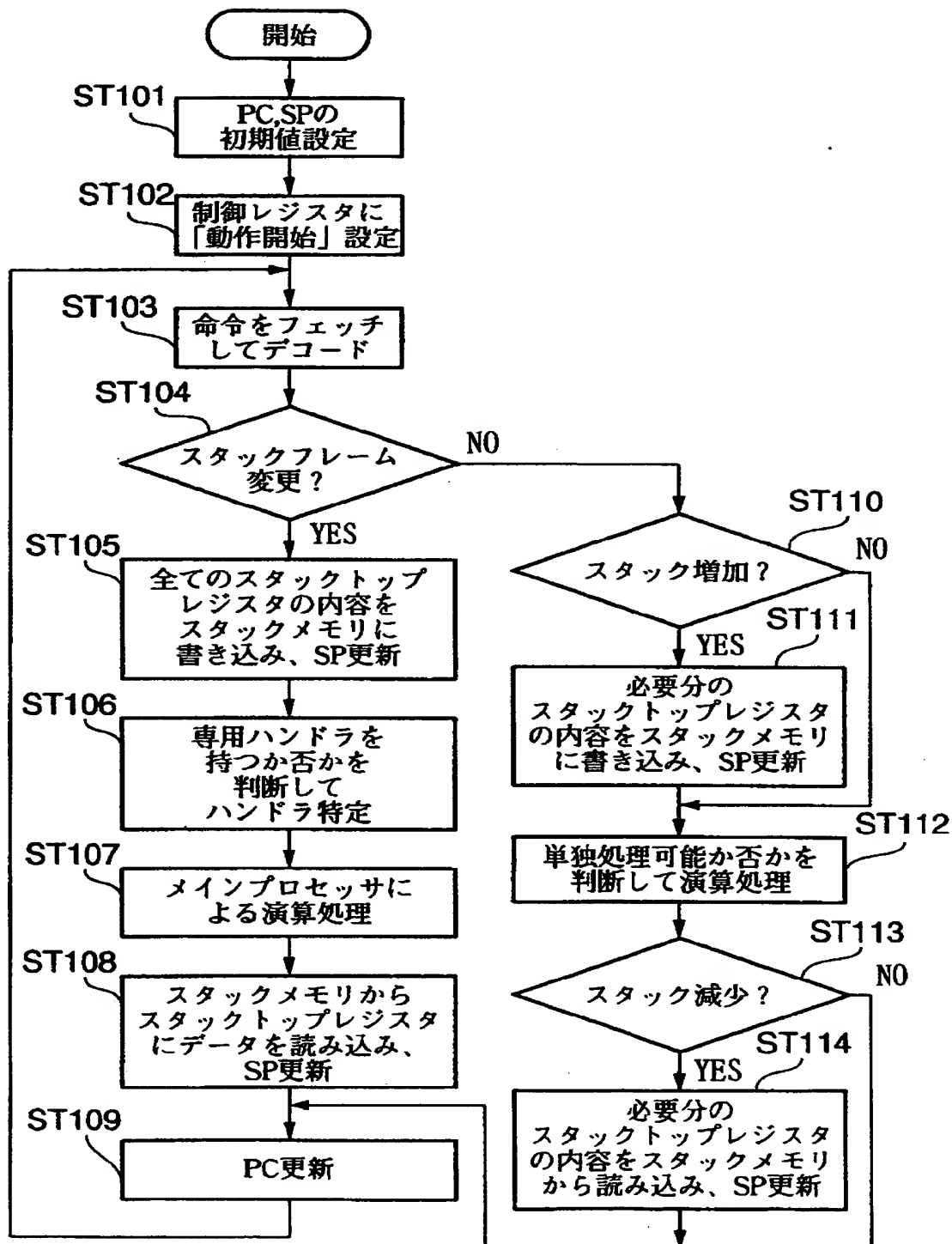
(a)

<演算後>

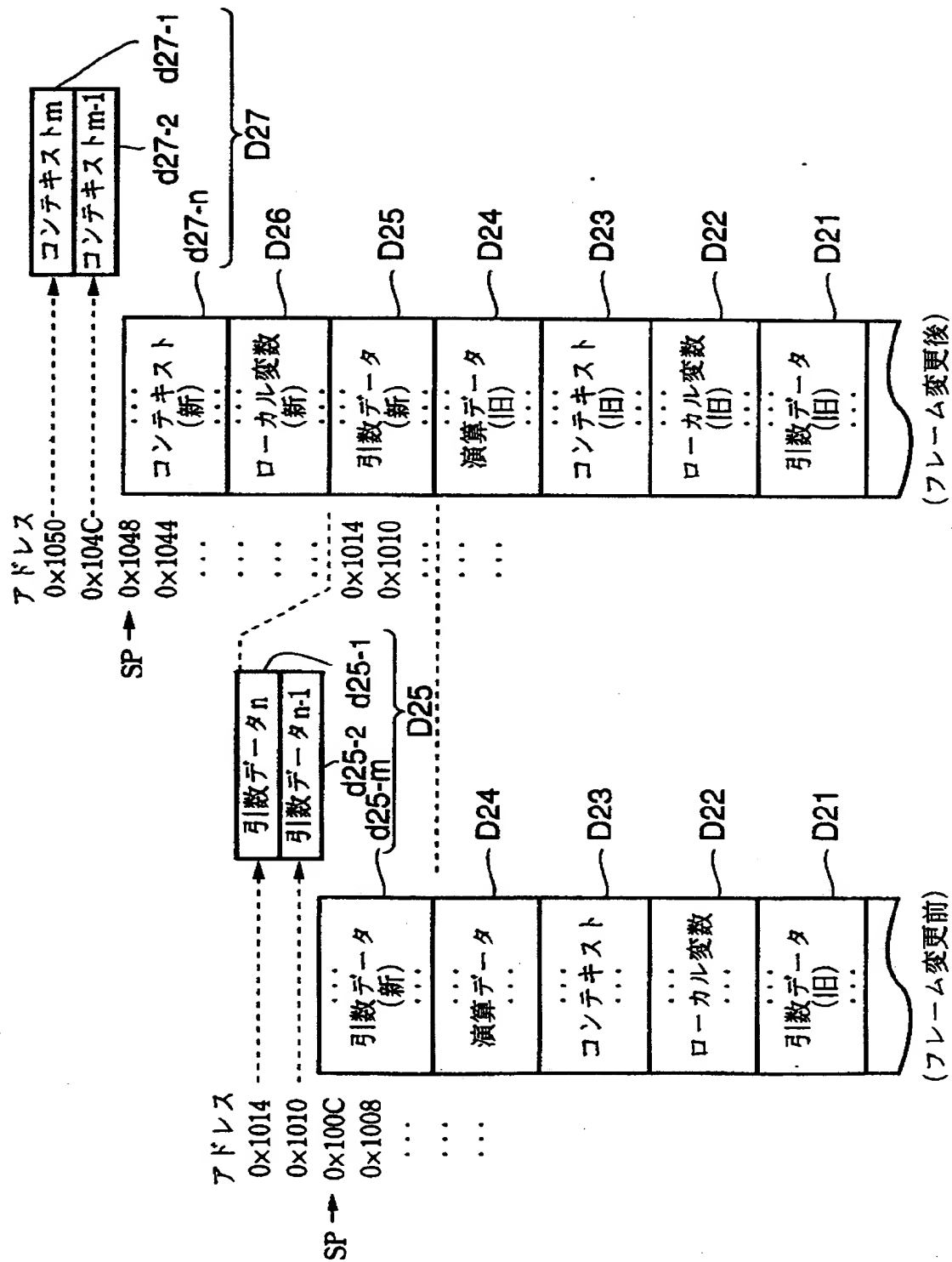


(b)

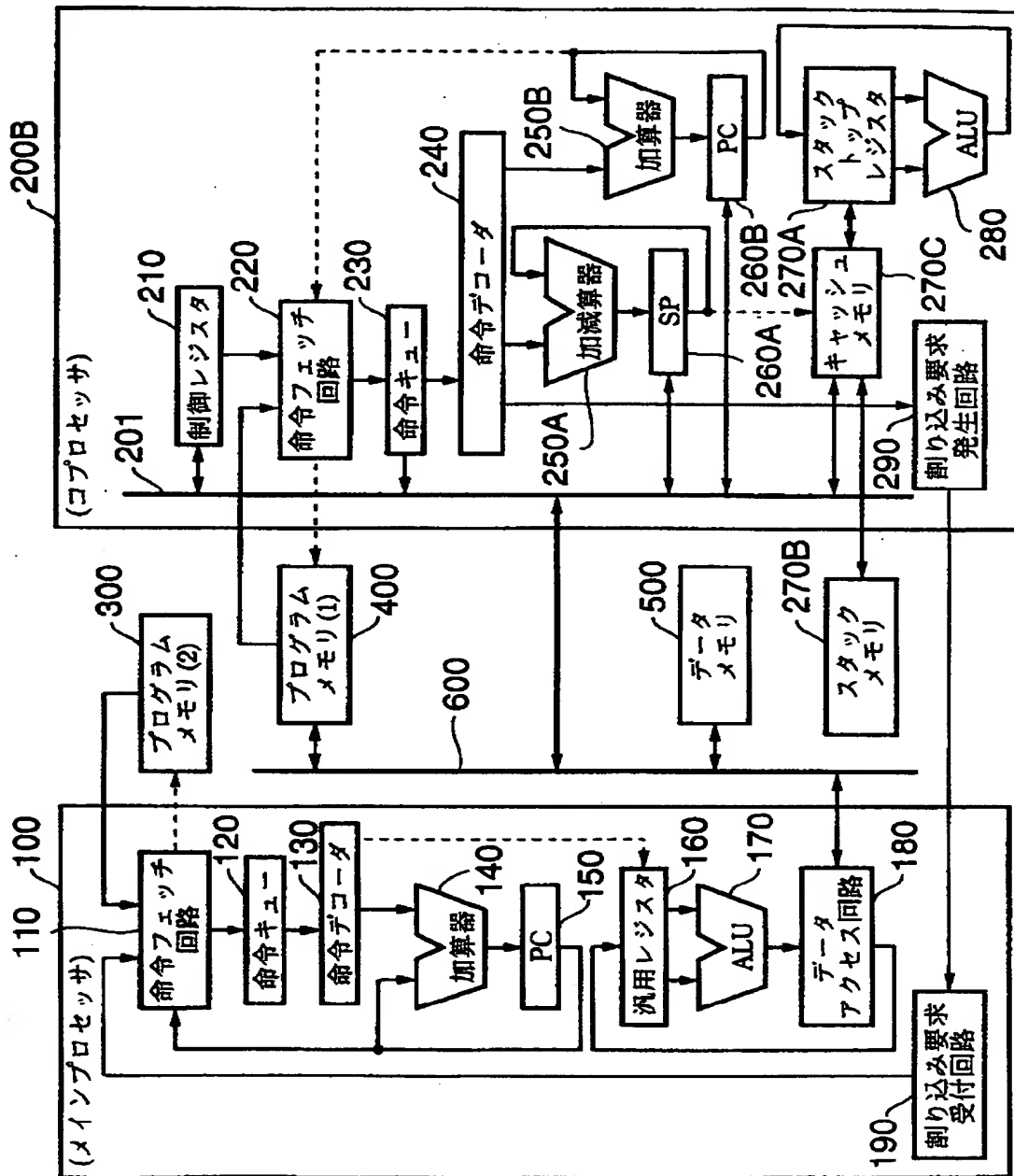
【図 9】



【図 1 0】

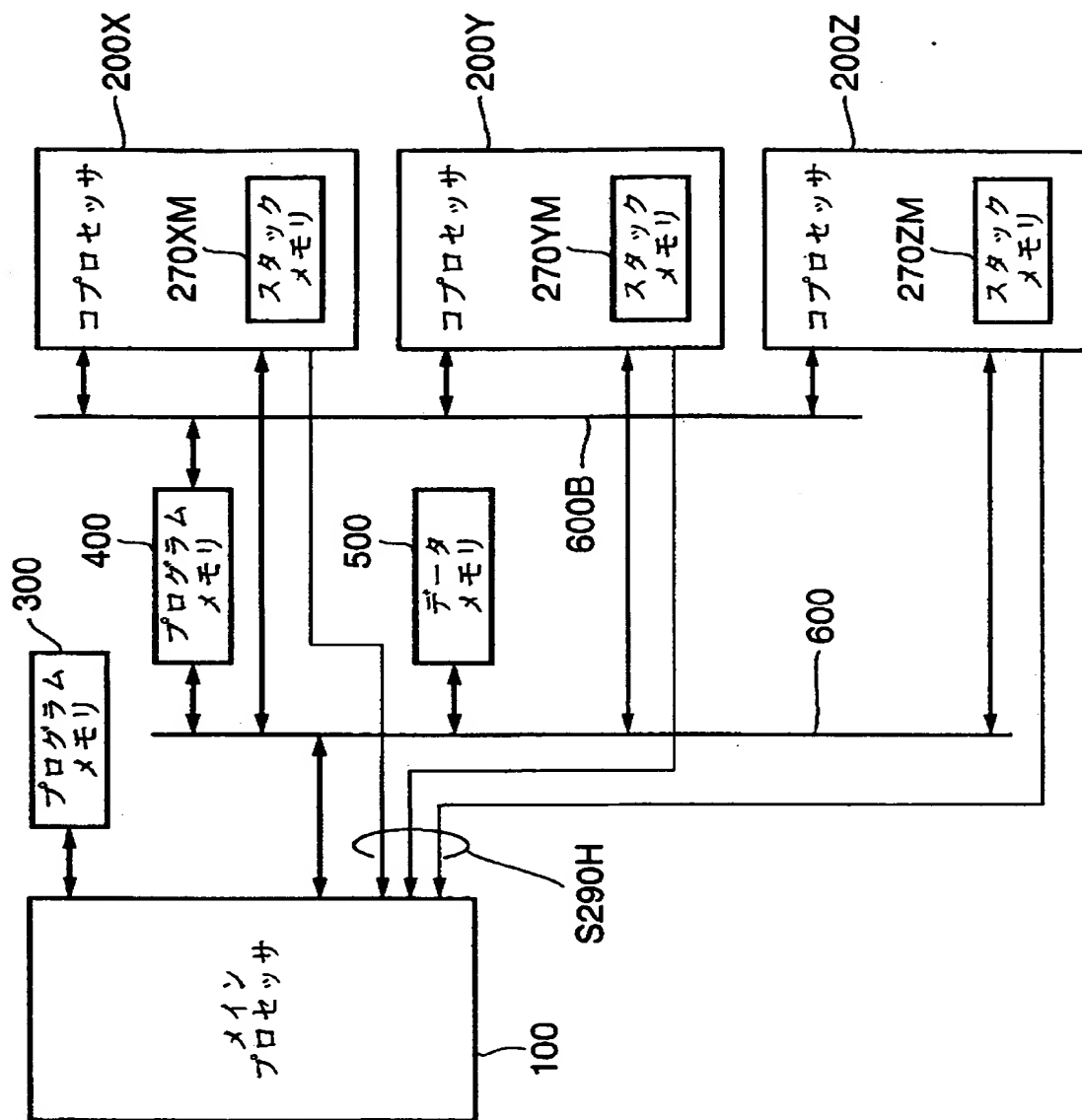


【図 1 1】





【図 1 2】



【書類名】 要約書

【要約】

【課題】 回路規模の増大を抑えつつ、特定の命令セットを高速に実行することが可能なマイクロプロセッサシステムを提供すること。

【解決手段】 プログラムに記述された命令を実行するマイクロプロセッサシステムにおいて、第1の命令セットをハードウェア上で実行すると共に第2の命令セットをソフトウェア上で実行するメインプロセッサ100と、前記メインプロセッサの管理下で動作して前記第2の命令セットをハードウェア上で実行するコプロセッサ200とを備える。前記コプロセッサ200は、前記第2の命令セットのうち、前記メインプロセッサ100の管理下にあるデータを操作する必要を生ずる特定の命令に遭遇した場合、前記メインプロセッサ100に通知を発行して当該命令の実行を依頼する。この場合、コプロセッサ200は、ハードウェア上でスタックポインタおよびプログラムカウンタを自ら更新する。

【選択図】 図1

認定・付加情報

特許出願の番号	平成11年 特許願 第331960号
受付番号	59901140671
書類名	特許願
担当官	坪 政光 8844
作成日	平成11年11月29日

<認定情報・付加情報>

【特許出願人】

【識別番号】	000004237
【住所又は居所】	東京都港区芝五丁目7番1号
【氏名又は名称】	日本電気株式会社

【代理人】

申請人	
【識別番号】	100108578
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所
【氏名又は名称】	高橋 詔男

【代理人】

【識別番号】	100064908
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所
【氏名又は名称】	志賀 正武

【選任した代理人】

【識別番号】	100101465
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所
【氏名又は名称】	青山 正和

【選任した代理人】

【識別番号】	100108453
【住所又は居所】	東京都新宿区高田馬場3丁目23番3号 ORビ ル 志賀国際特許事務所
【氏名又は名称】	村山 靖彦

出 願 人 履 歴 情 報

識別番号 [000004237]

1. 変更年月日	1990年 8月29日
[変更理由]	新規登録
住 所	東京都港区芝五丁目7番1号
氏 名	日本電気株式会社